TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TEIJO JUNTUNEN

PUUTAVARAN METSÄKULJETUKSEN OPTIMOINTI

Diplomityö

Tarkastaja: professori Risto Ritala

TIIVISTELMÄ

Teijo Juntunen: Puutavaran metsäkuljetuksen optimointi
Tampereen teknillinen yliopisto
Diplomityö, 38 sivua, 2 liitesivua
Syyskuu 2018
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Systeemien analysointi
Tarkastaja: professori Risto Ritala

Avainsanat: Reittioptimointi, kokonaislukuoptimointi, metaheuristiikka, toistuvat sovitukset

Kun hakkuukone on käynyt kaatamassa leimikolta puita pinoiksi, on seuraavaksi kuormatraktorin tehtävänä kerätä kyseiset pinot mahdollisimman tehokkaasti purkutien varteen purkulavoille jatkokuljetusta varten. Keruu tapahtuu harvesterin työskennellessä syntynyttä ajouraverkostoa pitkin. Tieto pinojen sijainnista ja ajourista tallennetaan GPS-paikannuksen avulla harvesterin työskentelydatasta. Tällä hetkellä metsätraktorin kuljettaja tekee keräyssuunnitelman kokemukseensa, koulutukseensa ja lahjakkuuteensa perustuen. Tämä aiheuttaa huomattavaa kuskien välistä vaihtelua keräyksen tehokkuudessa.

Tämän työn tarkoituksena oli luoda optimointialgoritmi, joka muodostaisi hyvän keräyssuunnitelman kuormatraktorinkuljettajan puolesta. Yhdessä metsäkonevalmistaja Ponsse oyj:n asiantuntijoiden kanssa luotiin vaatimusmäärittely, jonka mukaisesti algoritmin tulisi toimia. Tämän mukaan haluttiin, että osa algoritmille syötettävistä ajouraverkoston ajourista tulisi pystyä merkitsemään yksisuuntaisiksi. Myös ajourasegmenttien maksimiajokerrat täytyisi voida syöttää algoritmille ennen laskentaa. Laskenta-ajan haluttiin olevan alle 60 minuuttia.

Koska tehtävä on NP-vaikea kompleksisuusluokkaan kuuluvana mahdoton ratkaista täysin optimaalisesti, päätettiin optimoinnissa käyttää toistuvien sovitusten metaheuristiikkaa. Ensin muodostettiin GPS-datan perusteella suunnattu ja kustannuksilla painotettu karttagraafi, joka toimi herätteenä suunnitellulle optimointialgoritmille. Alustavaa ratkaisua lähdettiin parantamaan iteratiivisesti yhdistämällä edellisen ratkaisun reittejä toisiinsa. Yhdistäminen tapahtui joko liittämällä kahden reitin pinot toisiinsa satunnaisessa järjestyksessä tai vaihtamalla 1-3 pinoa kahden edellisen ratkaisun reitin välillä. Jos kumpikaan näistä parituksista ei ollut kustannuksellisesti tehokkaampi kuin aikaisemman ratkaisun reittien yhteiskustannus, merkittiin aikaisemman iteraation yhteiskustannus parituskustannukseksi. Näin syntyneestä sovituskustannusgraafista muodostetaan kaikki mahdolliset täydelliset sovitukset, joista etsitään halvin sovitus. Ratkaisun päätyessä lopulta johonkin ääriarvoon, puretaan satunnaisesti yksi tai useampia reittejä yksittäisiä pinoja sisältäviksi reiteiksi ja jatketaan optimointia.

Suunniteltua toistuvien sovitusten menetelmään perustuvaa algoritmia käytettiin ratkaisemaan Ponsse Forwarder Game:n pelikartan reititysongelmaa. Kyseistä karttaa oli käytetty ”Taitaja 2014” -kilpailussa metsäkoneenkuljettajaopiskelijoiden reittisuunnittelukyvyn arvioimiseen. Algoritmin suoritusta verrattiin yhdeksän Taitaja-kilpailun finalistin suorituksiin. Algoritmi suoriutui tuotoksella mitattuna keskitasoisesti, vaikka ajourarajoitettujen reittiosuuksien yhdistämisessä tehtävään oli tulosten perusteella havaittavissa puutteita.

ABSTRACT

Teijo Juntunen: Optimizing log collection at forest stand
Tampere University of Technology
Master of Science Thesis, 38 pages, 2 Appendix pages
September 2018
Master's Degree Programme in Automation Technology
Major: System Analysis
Examiner: Professor Risto Ritala

Keywords: Optimization, integer linear programming, repeated matching, metaheuristics

After harvester has felled trees into piles on a harvesting area, a forwarder must then transport the lumber to the dedicated transport area. The forwarder moves along the road network that harvester made during it's working phase. The pile location data is collected from the harvester's GPS. Currently the forwarder driver plans the collection routes using her personal expertise and experience. Therefore, there is a considerable amount of variance in the efficiency of the planned routes between different operators.

In this thesis, an optimization algorithm using repeated matching heuristic was used to optimize a set of routes for log collection at forest stand. The optimization result was then compared against results that forwarder driver students had had in Ponsse Forwarder Game. Ponsse Forwarder Game is a java-based game that is designed to help forwarder drivers improve their collection route planning.

The requirement analysis for the optimization algorithm was done together with the representatives of forest machine manufacturer Ponsse Ltd. They stated that some of the driving segments should be drivable only to one direction, some segments should only be drivable a pre-determined amount of times and calculation time should preferably not exceed 60 minutes.

Since the vehicle routeing problem complexity is NP-hard, it is impossible to solve the exact optimal solution as there are usually hundreds of piles to be collected in every forest stand. Therefore, a metaheuristic approach of repeated matching was used to get a good solution in allotted time.

First, a directed and cost-weighted graph based on GPS data was generated. Then algorithm was initialized with a feasible solution. Routes in the initial solution were paired with each other and the lowest cost perfect matching of the paired routes was chosen to be the next solution. Solution that had the lowest cost was saved and process was repeated as long as the cost of the solution improved. To avoid a local minimum, one or more routes from the solution is broken into smaller routes when there has not been any improvement in the solution for many iterations.

The performance of the repeated matching algorithm was studied as a function of calculation iterations and tested against the forwarder driver student performance in the Ponsse Forwarder Game. These students were the 9 finalists of "Taitaja 2014" competition for forest machine driver students in Finland. The RM-solution performed adequately against

these drivers even when there were noticable deficiencies in the way that the segments that had drivability limits were handled in this version of the algorithm.

ALKUSANAT

Tämä diplomityö on tehty Tampereen teknillisen yliopiston automaation ja hydrauliikan yksikössä. Kiitän professori Risto Ritalaa mielenkiintoisesta aiheesta, hyvin jäsennellystä ja kokemusta huokuvasta ohjauksesta sekä työmahdollisuuksista. Kiitän myös Ponsse Oyj:n edustajia ja heistä erityisesti Kalle Einolaa sekä Simo Tauriaista, jotka määrittelivät optimointialgoritmin vaatimukset, kuvasivat kuormatraktorin toimintaympäristön ja tarjosivat referenssidatan algoritmin toiminnan tarkasteluun.

Kiitokset myös TTY:lle työhuoneesta, työvälineistä ja työyhteisöstä sekä kaikesta siitä asiantuntevasta opetuksesta mistä opintojeni aikana olen saanut nauttia.

Lopuksi vielä kiitokset perheelleni kaikesta muusta.

Tampereella, 17.9.2018

Teijo Juntunen

SISÄLLYSLUETTELO

LYHENTEET

lyhenne	Määritelmä
GPS	Geographical Positioning System, satelliittipaikannusjärjestelmä
VRP	Vehicle Routeing Problem, ajoneuvon reititysongelma
CVRP	Capacitated Vehicle Routeing Problem, Kapasiteettirajoitetun ajoneuvon reititysongelma
RM, Repeated Matching	Toistuvien sovitusten menetelmä
PFG	Ponsse Forwarded Game, Ponssen tuottama javapohjainen peli, joka on tarkoitettu kuormatraktorinkuljettajien reittisuunnittelun harjoittamiseen.
CVRPMT	Capacitated Vehicle Routeing Problem with Multiple Trips, Kapasiteettirajoitetun ajoneuvon useiden ajokertojen reititysongelma.

1. JOHDANTO

Metsät ovat yhä Suomen tärkeimpiä luonnonvaroja. Tämän vuoksi on perusteltua pyrkiä jatkuvasti tehostamaan metsätaloudessa ja -teollisuudessa käytettyjä menetelmiä. Metsäkonevalmistaja Ponsse Oyj:n edustajien käytyjen keskustelujen mukaan tehokkain lähitulevaisuuden kehityskohde puunkorjuussa on puunkorjuureittien suunnittelun optimointi ja automatisointi.

Kun hakkuukone on tehnyt työnsä ja jättänyt metsään ajamien reittiensä varsille tekemänsä karsitut puupinot, on kuormatraktorin tehtävä kerätä kyseiset pinot määrätyn purku-uran keruulavoille. Tällä hetkellä kuormatraktorin kuljettaja tekee kokemukseensa, lahjoihinsa ja koulutukseensa pohjautuen tehokkaimman mahdollisen keräysreittisuunnitelman itse. Tehtävä on vaativa, sillä metsässä voi olla satoja kerättäviä pinoja ja useita eri puulajeja. Tällöin erilaisia kaikki pinot metsästä keräämiä reittiyhdistelmiä on käytännössä rajattomasti. Huomioon täytyy ottaa myös reittien kuljettavuus, mahdolliset ajettavat kulkusuunnat ja kuormatraktorin kuormauskapasiteetti. Kuljettajan on myös itse määritettävä mitä puulajia hän millekin keruulavalle purkaa. Tätä monimutkaista ongelmaa ratkaisemaan tarvitaan optimointialgoritmi, joka muodostaisi mahdollisimman hyvän reittisuunnitelman ja näin poistaisi kuljettajan suunnittelukyvykkyydestä aiheutuvan varianssin keräystehokkuudesta.

Toimiakseen optimointialgoritmi tarvitsee tietoa pinojen välisten siirtymien kustannuksista. Jatkuvasti kehittyvät GPS-paikannusjärjestelmät mahdollistavat ajouraverkoston kartan luomisen hakkuukoneen työskentelyn aikana. Näin voidaan kerätä tietoa metsään jääneiden pinojen sijainnista suhteessa toisiinsa. Karttadatasta voidaan muodostaa suunnattu sekä painotettu graafi, jota käytetään optimointialgoritmin perustana. Hakkuukoneenkuljettaja voi myös merkitä karttadataan harkintansa mukaan kulkusuunta- ja ajokertarajoitteita. Tulevaisuudessa kustannusdataa voidaan tarkentaa myös mahdollisesti ajourien kantavuusmittauksin [1]. Optimointituloksen hyvyyden kannalta kustannusten kuvaavuus on ensiarvoisen tärkeätä.

Vastaavanlaista ongelmaa on aiemmin tutkittu Ruotsissa [2] . Hakkuualueiden koot ovat Ruotsissa keskimäärin Suomea suurempia ja kyseinen tutkimus oli tehty yli tuhannen pinon leimikoille ilman kulkusuunta- tai ajokertarajoitteita. Tässä diplomityössä asiantuntijoina toimineet metsäkonevalmistaja Ponsse Oyj:n edustajat määrittelivät, että algoritmi tuli optimoida noin 200 pinon leimikoille, joille on määritelty ajourasegmenttien ajosuunta- ja ajokertarajoitteita. Pienemmillä hakkualueilla myös kuormien purku-uran liittymäreittien eriävät kulkukustannukset tulevat laskennallisesti merkittävämmiksi kuin suuremmilla leimikoilla. Näillä rajoitteilla pyrittiin approksimoimaan mahdollisimman

hyvin todellista suomalaisessa metsässä suoritettavaa keräystilannetta. Sallittu laskenta-aika määriteltiin korkeintaan 1 tunnin mittaiseksi.

Reittioptimointiongelmat, ovat yleensä kompleksisuudeltaan NP-vaikeita [3], jolloin täysin optimaalinen ratkaisu käy nopeasti graafin solmukohtien määrän lisääntyessä mahdottomaksi. Tällöin käytetään nk. metaheuristisia ratkaisumenetelmiä, jotka pyrkivät jollain heuristiikalla luomaan yhdestä mahdollisesta ratkaisusta iteratiivisesti parempia ratkaisuja. Yleensä päädytään johonkin paikalliseen optimiin, josta poistutaan valittuun heuristiikkaan perustuvalla tavalla ja jatketaan ratkaisua mahdolliseen parempaan paikalliseen optimiin.

Pääasialliseksi optimointimenetelmäksi valittiin tutkimuksissa [2] ja [4] käytetty toistuvien sovitusten menetelmä (Repeated Matching). Algoritmin ratkaisu tuottaa kokoelman reittejä, jotka annettuja rajoitteita noudattaen keräävät jokaisen pinon karttagraafia vastaavalta leimikolta. Ratkaisua parannetaan muokkaamalla edellisen iteraatiokierroksen ratkaisun reittejä, joko yhdistämällä kahden reitin pinot yhdeksi reitiksi, vaihtamalla osa reittien pinoista satunnaisesti kahden reitin välillä tai säilyttämällä aiemmat reitit. Reittipari sovitetaan aina niin, että sovituskustannus on pienin mahdollinen. Kaikista mahdollisista reittien sovituskombinaatioista etsitään kokonaislukuoptimoinnilla halvin täydellinen sovitus.

Työn rakenne on seuraava: Aluksi kuvaamme kuormatraktorinkuljettajan työkalut ja toimintaympäristön tämän optimointiongelman erityisvaatimuksien selvittämiseksi. Toiseksi käydään läpi käytettyjen ratkaisumenetelmien teoria yleisesti, minkä jälkeen kuvataan käytetyn optimointialgoritmin rakenne tarkemmin. Seuraavaksi kerrotaan, kuinka suunnitellun optimointialgoritmin hyvyyttä ja tehokkuutta mitataan. Tuloksen kehittymistä tarkastellaan algoritmin iteraatiokertojen funktiona. Ratkaistun reittikokoelman hyvyyttä testataan ihmiskuljettajien suorituskykyä vastaan todellista tilannetta mukailevassa reittisuunnittelutehtävässä. Lopuksi esiteltyjen tulosten perusteella tehdään johtopäätökset algoritmin toiminnasta.

.

2. KUORMATRAKTORIN TOIMINTAYMPÄRISTÖ

Luvussa kuvataan lyhyesti puutavaran metsäkuljetuksen historiaa, sekä kuormatraktorin ja sen kuljettajan nykyinen toimenkuva sekä toimintaympäristö. Samalla tutustutaan kuormatraktorin ja sen toimintaympäristön aiheuttamiin rajoitteisiin, sekä puutavaran keräämiseen liittyvään ammattisanastoon. Näiden tietojen tavulla voidaan perustella siirtymäkustannusten muodostumista, optimointialgoritmin painotuksia ja tulevaa tarvetta täysin automatisoiduille keräyssuunnitelmille.

2.1 Puutavaran metsäkuljetuksen historia ja kehitys

Vielä 1940-luvulla puut kaadettiin pääasiallisesti käsisahoilla ja tärkein kuljetusväline puiden kuljettamiseen metsästä oli hevosen vetämä reki. Traktoreita oli alettu testata metsäkuljetustarpeisiin Pohjois-Amerikassa jo 1920-luvulta eteenpäin, mutta ne yleistyivät hitaammin kuin maataloudessa. Neuvostoliitossa ja Pohjois-Amerikassa 50-luvun lopussa kehitetyt ohjaustehostetut metsätraktorit alkoivat saada jalansijaa metsätaloudessa seuraavalla vuosikymmenellä [5].

Puun kaadossa oli siirrytty 60-luvulle tultaessa kokonaan moottorisahaan, jota pystyi jo tuossa vaiheessa käyttämään yhden henkilön voimin. Samaan aikaan kehitettiin koneellista puunkaatoa.

Pohjoismaat seurasivat muutaman vuoden viiveellä suurempia metsätalousmaita. Ensimmäinen ruotsalainen erikoistunut metsätraktori tuli myyntiin 1957 ja hydraulisella tarraimella varustettu versio kaksi vuotta myöhemmin. Suomessa valmistettu ensimmäinen monitoimihakkuukone nimeltä Pika 50 tehtiin Rosenlewin konepajalla vuonna 1968 ja Kuva 1 nähtävä Einari Vidgrenin suunnittelema ensimmäinen Ponsse-kuormatraktori rakennettiin vuonna 1969 [6].

Ruotsissa suoritettiin 80% puiden metsäkuljetuksista hevosella vielä vuonna 1960. Kymmenessä vuodessa tilanne oli kääntynyt, sillä vuonna 1970 jo 95% puista kuljetettiin metsästä koneellisesti. Vuoteen 1972 mennessä kaikki hakkuuprosessin osa-alueet oli jo koneellistettu.

Kuva 1. *Ponsse Dino -metsätraktori vuodelta 1969.* ***[7]***

Metsäkoneiden perusperiaatteet ovat pysyneet samoina 1990-luvulta asti. Kehitystä on tehty lähinnä tuottavuuden parantamiseksi ja kustannusten minimoimiseksi. Metsäkoneenkuljettajan ympärillä tapahtuvan jatkuvan kehityksen vuoksi ihminen uhkaa jäädä tehokkuuden pullonkaulaksi. Tämä lisää tulevaisuudessa tarvetta täysin automatisoituun puunkorjuuseen.

2.2 Kuormatraktori

Kuten kuvasta 2 nähdään, kuormatraktorin pääosat ovat traktori, kuormain ja kuormatila. Traktori kulkee leimikolla yleensä n. 8 km/h nopeudella, paitsi jos seuraava kerättävä pino on alle 15 metrin päässä edellisestä pinosta - jolloin kuljetaan n. 4 km/h nopeudella. Hyvä näkyvyys kopista on tärkeätä, joten joka suuntaan lasitetun kopin lisäksi kuskilla on käytettävissään monitoreja, jotka näyttävät kamerakuvaa sokeista kulmista. Muista monitoreista voidaan seurata myös reittisuunnitelman etenemistä ja GPS:n antamaa karttasijaintia.

Kuormaimen nosturin kääntökulma on 360° ja ulottuma vaihtelee tyypistä riippuen n. 8-10 metriin. Kuormatilan kapasiteetti riippuu kuormatraktorin mallista, mutta on yleensä kymmeniätuhansia kiloja.

Kuva 2. *Kuormatraktori leimikolla* ***[8]***

2.3 Kuormatraktorin kuljettajan toiminta hakkuualueella

Ennen kuormatraktorin (metsätraktorin, ajokoneen) tuloa hakkuualueelle eli leimikolle, hakkuukone (monitoimikone, harvesteri) on käynyt karsimassa ja kaatamassa työmääräyksen perusteella valitut puut alueelta. Nämä puut kasataan pieniin puutavaralajikohtaisiin pinoihin (sormipinoihin) kuljettajan suunnitteleman ajouran varrelle. Kuormatraktorin tehtävänä on kerätä kyseiset pinot ajouria pitkin leimikolta tien varteen välivarastolle omiin puutavaralajikohtaisiin kasoihinsa. Välivarasto sijaitsee yleensä 100-500 metrin päässä hakkuualueesta [9]

Tällä hetkellä kuormatraktorin kuljettaja suunnittelee itse keräysreittinsä omaan koulutukseensa, lahjakkuuteensa ja työkokemukseensa nojautuen. Lähtötiedot, kuten leimikon kartat, harvesterin kulkemat ajourat ja kerättävien puupinojen sijainnit ovat nykyisin sähköisessä muodossa kuljettajan nähtävillä kuormatraktorin hytin monitoreista. Kuva 3 nähdään nykyaikainen kuormatraktorin kuljettajan työpiste.

Kuva 3. *Kuormatraktorin hallintalaitteistoa* ***[10]***

GPS-paikannusjärjestelmien kehitys mahdollistaa yhä tarkemman keräysreittisuunnittelun sekä laskennallisen reittioptimoinnin, kun tieto leimikon ajourista ja sormipinojen sijainneista saadaan nopeasti laskennan herätedataksi. Tällä hetkellä, kun kuormatraktorinkuljettaja suunnittelee ajoreittiä, on hänen otettava huomioon maanpinnan muotojen vaihteluiden asettamat haasteet keräämiselle sekä säiden ja ajourapohjan kulkukelpoisuuden vaikutukset ajourien ajettavuuteen.

Ajon aikana kuormatraktorin kuljettaja reagoi mahdollisiin muuttuviin olosuhteisiin ja etukäteistiedoista poikkeaviin tilanteisiin sekä arvioi kuhunkin tilanteeseen mielestään sopivimman kulkusuunnan ja -nopeuden, parhaimman koostumuksen kuormalle ja sopivimmat tulosuunnat kohtiin, joista keräys suoritetaan.

2.4 Kuormatraktorin kuljettajan toiminnan vaikutus keräyksen tehokkuuteen

Ponsse Oyj:n edustajat arvioivat, että kuormatraktorin kuljettajan henkilökohtaisen taidon vaikutus keräysaikaan on merkittävä. Tätä johtopäätöstä tukee myös ”Taitaja 2014” -kilpailussa käytetyn Ponsse Forwarder Game -reittisuunnittelusimulaattorin raportoimat tulokset, jossa kuormatraktorinkuljettajiksi opiskelevien suunnittelemien korjuureittien tuotto oli keskimäärin 24.5 m^3/h. Parhaimman tulos oli 13.8% keskiarvotuottoa parempi ja huonoimman tulos n. 10% keskiarvoa huonompi. Tulokset on kerätty Taitaja-kilpailun finaalista, johon on päässyt eri puolella Suomea järjestetyissä alkukarsinnoissa parhaiten pärjänneet. Näin voidaan olettaa, että suoritukset ovat jonkin verran keskimääräistä opiskelijaa parempia. Tämän takia pidettiin perusteltuna ja hyödyllisenä vertailla tässä työssä

suunnitellun optimointialgoritmin suorituskykyä mainitun Taitaja-kilpailun finaalistien PFG:ssa saamiin tuloksiin. Kilpailutulokset ovat nähtävissä

Myös aiemmat Metsäntutkimuslaitoksen tutkimukset, joissa verrattiin kuormatraktorin-kuljettajan suunnittelemaa ajoreittiä reittioptimoitimenetelmillä muodostettuun ajoreittiin, antoivat viitteitä siitä, että optimoinnilla voidaan parantaa kuljettajien kokonaisajosuoritetta jopa yli 10% [11].

3. REITTIOPTIMOINTIMENETELMÄT

Tässä luvussa käydään läpi reittioptimointimenetelmiä (Vehicle Routeing Problem, VRP), tätä työtä koskevia reittioptimoinnin alalajeja kuten kapasiteettirajoitetun ajoneuvon reittioptimointiongelma (Capacitated Vehicle Routeing Problem, CVRP) ja Dijkstran algoritmi. Yleisen käsittelyn jälkeen esitellään tarkemmin työssä käytetyt algoritmit ja perustellaan juuri niiden soveltuvuus tähän työhön.

3.1 Laskennallinen kompleksisuus

Useimmat kiinnostavista reittioptimointiongelmista ovat kompleksisuusteorian joukkoopin mukaan NP-vaikea -luokkaan kuuluvia tehtäviä [3]. NP tarkoittaa tehtäväjoukkoa, jotka ovat ratkaistavissa ei-deterministisellä Turingin koneella polynomiaalisessa ajassa. NP-vaikeat tehtävät ovat puolestaan sellaisten tehtävien joukko, jotka ovat laskennallisesti vähintään yhtä vaikeita, kuin NP-luokan vaikeimmat tehtävät. Käytännössä useiden reittioptimointiongelmien mahdollisten ratkaisujen määrä kasvaa eksponentiaalisesti ongelman solmukohtien lukumäärän funktiona. Tämä estää ongelman tarkan globaalin optimiratkaisun löytämisen jo melko pienissä reittioptimointitehtävissä.

Yksinkertaisemmat ongelmat, kuten lyhimmän etäisyyden määrittäminen graafin kahden solmukohdan välille, kuuluvat polynomiaalisessa ajassa ratkaistavien tehtävien luokkaan eli P-luokkaan. Näiden ns. helppojen ongelmien ratkaisuun voidaan käyttää myös globaalin minimikustannuksen määrittäviä algoritmeja, kuten Dijkstran algoritmi [12]tai A*-algoritmi [13].

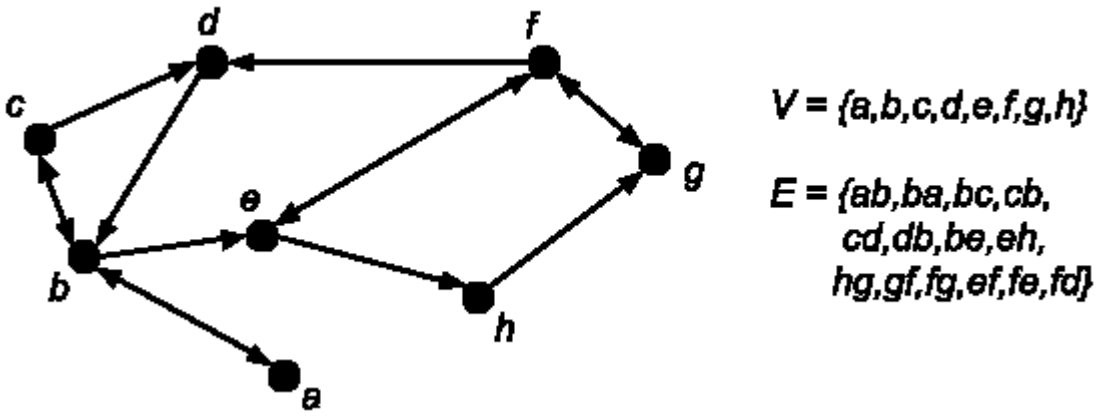
Optimaalisen ratkaisun löytymisen epätodennäköisyys, kun ongelman solmukohtien määrä kasvaa, on pakottanut tutkimaan erilaisia metaheuristiikkoja. Näissä pyritään ensin löytämään jonkinlainen alustava ratkaisu tehtävälle, jota ryhdytään kyseisen heuristiikan mukaisesti hiomaan iteratiivisesti paremmaksi ratkaisuksi. Jos algoritmi suppenee johonkin paikalliseen optimiin, tallennetaan tämä optimin arvo ja hajotetaan ratkaisu johonkin aikaisempaan tilaan, joka on riittävän kaukana kyseisestä optimista. Tämän jälkeen iteratiivinen prosessi lähtee jälleen parantamaan ratkaisua. Koska heuristiikat sisältävät usein tarkoituksellisesti satunnaisuutta, toivotaan algoritmin päätyvän nyt aikaisemmasta poikkeavaan paikalliseen optimiin. Tällä tavoin ratkaisu paranee iteraatiokertojen funktiona.

3.2 Graafiteorian terminologiaa

Kartat kuvataan reittioptimointitehtävissä yleensä Kuva 4 kaltaisina graafeina. Graafit auttavat havainnollistamaan reititysongelman rakennetta algoritmin suunnitteluvaiheessa ja antavat standardin ongelman ratkaisemiseksi useilla eri menetelmillä. Ne ovat myös vaivattomia muuttaa vastaavan informaation sisältäviksi matriiseiksi algoritmia ohjelmoitaessa.

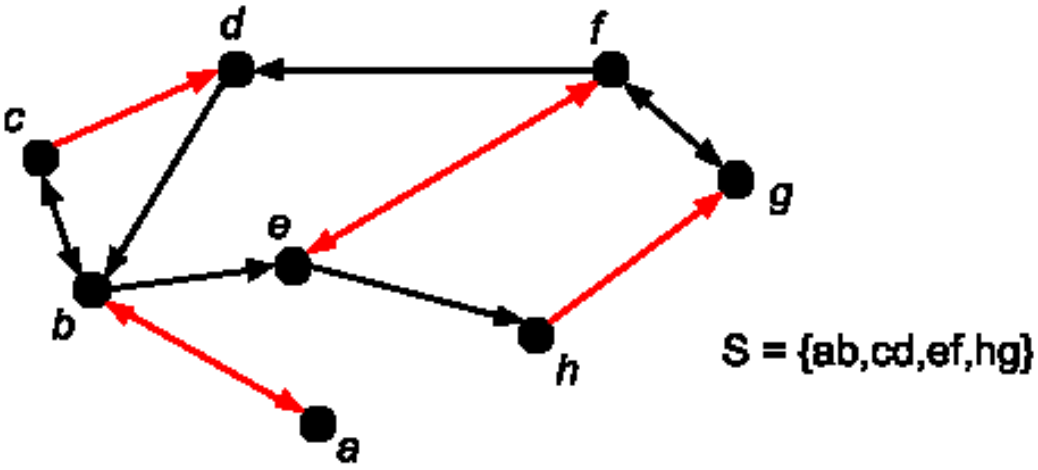
Graafi $G = (V, E)$ koostuu *solmujen* (vertex) joukosta V ja solmuparien muodostamasta *särmien* (edge) joukosta E. Joukko E on ns. multijoukko, eli sen alkiot voivat esiintyä joukossa useita kertoja [14]. Tässä työssä graafeilla kuvataan kuormatraktorin käyttämää tieverkostoa risteyksineen ja kerättävine puupinoineen sekä eri keräysreittien halvimpien yhdistämiskustannusten muodostamaa verkostoa.

Tämän työn hakkuualueen karttagraafi on suunnattu ja painotettu. Sen solmut ovat puupinojen keruukohtia. Särmät puolestaan kuvaavat solmujenvälisiä yhteyksiä – solmuparin solmujen järjestys on merkitsevä. Kuhunkin särmään liitetty paino on siirtymästä aiheutuva kustannus. Kuva 4 nähdään esimerkki suunnatusta graafista. Nuolet kertovat siirtymien mahdolliset suunnat. Hakkuualueen kartassa suurin osa yhteyksistä on kaksisuuntaisia ja symmetrisiä, eli tällöin siirtymäkustannus kahden solmun välillä on sama molempiin suuntiin. Osa reiteistä on kuitenkin ajokelpoisia vain yhteen suuntaan, joka aiheuttaa myös epäsymmetriaa siirtymäkustannuksiin.



Kuva 4. *Esimerkki suunnatusta graafista.*

Graafin *sovitus* S on sellainen joukko särmiä joukosta E, etteivät mitkään joukon S särmät ole *vierekkäisiä*. Särmät ovat vierekkäisiä, jos niillä on yhteinen solmu. Sovitus on formaalisti *täydellinen*, jos kaikki graafin solmut ovat sovitettuja eli jokainen graafin solmu esiintyy sovituksen jossakin särmässä [14]. Kuva 5 on yksi mahdollinen täydellinen sovitus kuvan 4 graafille.



Kuva 5. *Eräs täydellinen sovitus kuvan 4 graafille.*

Tällä määritelmällä täydellinen sovitus on mahdollinen vain, jos solmuja on parillinen määrä. Koska solmuja voi ratkaisussa olla myös pariton määrä, on tässä työssä mahdollistettu myös solmun sovitus itsensä kanssa. Käytännössä solmu jää tällöin parittamatta. Toistuvien täydellisten sovitusten menetelmässä käytetyssä sovituskustannusgraafissa solmut ovat kokonaisia keräysreittejä purkupaikalta kerättävien pinojen kautta takaisin purkupaikalle. Sivut puolestaan kuvastavat onko reittien yhdistäminen mahdollista ja painotus kertoo kyseisen sovituksen kustannuksen. Tämän sovituskustannusgraafin ja siitä tehtävän sovituskustannusmatriisin muodostaminen käydään tarkemmin läpi kappaleessa 5.3.

3.3 Kapasiteettirajoitetun ajoneuvon reititysongelma

Ajoneuvon reititysongelma, VRP [15], pyrkii ratkaisemaan nopeimman tai pienimmän kulkukustannuksen reitin lähtöpisteen ja loppupisteen välille tunnetun kustannusgraafin, esim. tiekartan, avulla. Graafin tulee sisältää tieto solmukohtien välisistä yhteyksistä ja siirtymien aiheuttamista kustannuksista. VRP-ongelmia ovat esimerkiksi kaupunkien välisten lyhimpien reittien määritys. Tällöin graafin solmuja ovat kaupungit ja särmiä kaupunkien väliset yhteydet. Särmien painot eli siirtymäkustannukset voivat muodostua vaikkapa kaupunkien välisistä etäisyyksistä tai maksiminopeuksilla kuljetuista siirtymäajoista.

VRP-ongelmasta on olemassa useita käytännön tarpeista syntyneitä variaatioita. Teollisuudessa ollaan usein kiinnostuneita logistisista ongelmista: hyödykkeiden keräilyistä, siirroista ja keräysten tai siirtojen ajoituksista [16], [17]. Näiden tehtävien mukaan on nimetty joukko reititysongelmia. Puulajien keräilyssä oleellinen seikka on kuormatraktorin kapasiteetti, jota ei voida ylittää. Tällöin kyseessä on kapasiteettirajoitetun ajoneuvon reititysongelma, CVRP [18] ,

CVRP-kategorian ongelmissa on yksi keräilypiste, josta kaikki reitit alkavat ja joihin ne päättyvät. Tämän lisäksi jokaisessa solmussa käydään ainoastaan kerran, eikä ajoneuvon kapasiteettia saa koskaan ylittää. Näiden rajoitteiden sisällä pyritään muodostamaan mahdollisimman edulliset reitit lopulliseen ratkaisuun.

Koska käytettävissä on vain yksi ajoneuvo, joka joutuu tekemään kapasiteettirajoituksensa vuoksi todennäköisesti useampia keräys- ja purkukierroksia, on kyseessä toisaalta myös kapasiteettirajoitetun ajoneuvon useiden ajokertojen reititysongelma (Capacitated Vehicle Routeing Problem with Multiple Trips, CVRPMT) [19].

Tämän työn CVRPMT-ongelmassa ratkaistaan kahta sisäkkäistä optimointiongelmaa. Toisaalta kaikkien reittien yhteiskustannuksen täytyy olla mahdollisimman pieni ja toisaalta myös valittujen reittien sisäisten pinojärjestysten tulee olla optimaalisia. Edellinen tehtävä ratkaistaan toistuvien sovitusten menetelmällä ja jälkimmäinen kauppamatkustajan ongelman ratkaisulla. Molemmat formuloidaan kokonaislukuoptimointitehtäviksi.

3.4 Kokonaislukuoptimointi

Tässä työssä käytetään pääasiallisena optimointimenetelmänä toistuvien täydellisten sovitusten menetelmää ja reittien sisäisten järjestysten optimoinnissa myös kauppamatkustajan tehtävän ratkaisua. Kokonaislukuoptimoinnilla pyritään löytämään kustannuksellisesti tehokkain sovitus edellisen iteraation ratkaisun reiteille.

Puhtaassa kokonaislukuoptimointiongelmassa kaikki valintamuuttujat ovat kokonaislukuja, kun taas sekoitetussa kokonaislukuoptimointitehtävässä mukana voi olla myös jatkuvia valintamuuttujia. Puhtaan kokonaislukuoptimointiongelman perusmuotoilu on seuraava [20]:

$$\begin{aligned} &\min_{\{x_j\}} \sum_{j=1}^{n} c_j x_j \ , \\ &kun \sum_{j=1}^{n} a_{ij} x_j = b_i \quad (i = 1,2,\dots,m), \\ &x_j \geq 0 \qquad (j = 1,2,\dots,n), \\ &x_j \epsilon \, \mathbb{Z} \qquad (\forall j = 1,2,\dots,n) \end{aligned} \tag{1}$$

missä minimoitavaa funktiota kutsutaan kohdefunktioksi, jonka ratkaisuavaruutta supistetaan tehtävän määrittelemillä rajoiteyhtälöillä. Rajoitteina voidaan käyttää myös epäyhtälöitä, jolloin muotoiluun lisätään epäyhtälörajoitteet:

$$\sum_{j=1}^{p} a_{2ij} x_j \leq b_{2i} \qquad (i = 1,2,\dots,k) \tag{2}$$

Usein käytetään kokonaislukumuuttujan sijaan binääristä valintamuuttujaa x_j. Toistuvien sovitusten menetelmässä valintamuuttujat valitsevat sovituskustannusgraafin optimaalisen sovituksen.

Tämän työn ratkaisijana käytetty Matlabin kokonaislukuratkaisija [21] pienentää tehtävän ratkaisuavaruutta ensin annettujen yhtälö- ja epäyhtälörajoitteiden mukaisesti [22]. Tämän jälkeen se ratkaisee annetun tehtävän ilman kokonaislukurajoitetta eli jatkuva-arvoisena relaksaationa esimerkiksi Simplex-algoritmilla [23]. Ratkaisija lisää myös useita leikkauksia ratkaisuavaruuteen, joilla jatkuva-arvoisia ratkaisuja saadaan lähemmäs kokonaislukuarvoja. Näitä menetelmiä käytetään nopeuttamaan Branch and bound -algoritmia, joka pyrkii supistamaan jatkuva-arvoiset ratkaisut parhaimpaan kokonaislukuratkaisuun [24]. Samaa ratkaisijaa käytetään sekä toistuvien sovitusten menetelmässä, että kauppamatkustajan tehtävän ratkaisemisessa.

3.4.1 Toistuvien sovitusten kokonaislukuoptimointimenetelmä

Toistuvien sovitusten optimointimenetelmää on käytetty hyvin tuloksin esimerkiksi linja-autojen aikatauluttamiseen [25], vanhustenhoidon henkilökuntaresurssien optimointiin [16] sekä reittioptimointiongelmiin [26], [27]. Seuraavaksi kuvataan menetelmän periaatteet ja myöhemmin luvussa 4 esitetään tarkemmin, kuinka menetelmää on sovellettu tämän työn algoritmissa.

Joukon $A = \{a_1, a_2, \dots, a_n\}$, missä n on parillinen, täydellinen sovitus on kyseessä silloin, kun jokainen elementti $a_i \in A$ on sovitettu yhden ja vain yhden elementin $a_j \in A(j \neq i)$ kanssa. Toisin sanoen parillisen joukon A täydellinen sovitus jakaa sen sovitettuihin elementtipareihin. Tässä työssä kyseiset elementit ovat kuormatraktorin keräysreittejä eli kerättävistä puupinoista järjestettyjä jonoja.

Sovitetut reitit sisältävät molempien reittien pinot, joko yhdellä yhdistetyllä reitillä tai kahdella satunnaisesti uudelleenjärjestetyllä reitillä. Koska reittejä voi käytännössä olla myös pariton määrä, laajennetaan sovitusmääritelmää siten, että reitit voidaan sovittaa myös itsensä kanssa. Tällöin nämä reitit säilyttävät pinosisältönsä ennallaan. Tämä yleisempi minimisovitusongelma muotoillaan kokonaislukuoptimointiongelmaksi seuraavasti:

$$
\begin{aligned}
&Minimoidaan\ z = \sum_{i=1}^{k}\sum_{j=1}^{k} d_{ij}z_{ij} \\
&kun \sum_{j=1}^{k} z_{ij} = 1, \quad i = 1, \dots, k \\
&\sum_{\mathrm{i}=1}^{k} z_{ij} = 1, \quad j = 1, \dots, k \\
&z_{ij} = z_{ji}, \qquad i, j = 1, \dots, k \\
&z_{ij} \in \{0,1\}, \quad i, j = 1, \dots, k
\end{aligned}
\tag{3}
$$

missä d_{ij} -elementit ovat sovituskustannusmatriisi *D*:n elementit. Binäärinen valintamuuttuja z_{ij} saa arvon 1 kun sitä vastaava sovituskustannus d_{ij} on valittu ratkaisuun ja arvon 0 silloin kun kustannusta ei ole valittu. On huomioitava, että symmetrisyysehdon vuoksi ratkaisuun sisältyy aina d_{ij}-kustannuksen vastinpari d_{ji}, mutta itsensä kanssa sovitetun reitin kustannus d_{ii} on ratkaisussa vain kerran. Tämän vuoksi d_{ii}-kustannuksen täytyy olla kaksinkertainen todelliseen arvoonsa verrattuna, jotta sen paino olisi verrannollinen kahden eri reitin sovituksen kustannuksen kanssa.

3.4.2 Kauppamatkustajan ongelman ratkaisu

Kun kahden reitin pinosisällöt yhdistetään toistuvien sovitusten menetelmässä satunnaisesti, voi välillä syntyä pienen kustannuksen ratkaisuja, joissa saatetaan kuitenkin tehdä tarpeettomia edestakaisia liikkeitä ja silmukoita. Tämä voidaan välttää korjaamalla välillä kokonaisratkaisun reittien pinojen sisäinen järjestys kokonaislukuoptimoinnilla mahdollisimman edulliseksi. Kyseinen ongelma tunnetaan myös kauppamatkustajan ongelman nimellä [28].

Kauppamatkustajan ongelmassa graafin solmukohdat ovat kaupunkeja, joissa kauppamatkustajan tulisi käydä kerran ja vain kerran, ennen kuin hän palaa takaisin kotikaupunkiinsa. Graafin sivujen painot määrittelevät kaupunkien väliset siirtymäkustannukset. Metsätraktorin reititystehtävässä yhden reitin sisältämät pinot vastaavat kaupunkeja ja sivujen painot ovat pinojen välisiä siirtymäkustannuksia. Ongelma formuloidaan kokonaislukuoptimointiongelmaksi yhtälön (4) mukaisesti:

$$Minimoidaan \sum_{i,j \in E} d_{ij} x_{ij}$$

$$kun \sum_{j \in V} x_{ij} = 2 \quad \forall i \in V$$

$$\sum_{i,j \in S, i \neq j} x_{ij} \leq |S| - 1 \quad \forall S \subset V, S \neq \emptyset \tag{4}$$

$$x_{ij} \in \{0,1\}$$

missä x on valintamuuttuja, jolla valitaan tehtävän kaikkien solmujen joukosta V solmujen i ja j välinen sivu ratkaisuun siten, että jokaisesta solmusta j on yhteys tasan kahteen muuhun solmuun joukosta V. Näin mahdollistuvat kuitenkin alireittisilmukat, jolloin ratkaisu ei muodosta vain haluttua yhtä reittiä. Tämä estetään yhtälön (4) rajoiterivillä 3, jossa jokainen joukon V epätyhjä aito osajoukko S määrätään sisältämään yhdistäviä sivuja x korkeintaan $|S|$-1 kappaletta. Näin formuloidun kauppamatkustajan tehtävän ratkaisuun käytetään Matlab-ohjelmiston dokumentaatiossa kuvattua menetelmää [29]. Kyseinen menetelmä käyttää ohjelmiston omaa kokonaislukuratkaisijaa, jota käytetään myös toistuvien sovitusten menetelmässä.

Koska kauppamatkustajan ongelma on myös tunnetusti laskennallisesti NP-vaikea tehtävä [28], käytetään tätä kokonaislukuoptimointiin perustuvaa ratkaisumenetelmää vain harkitusti laskennan edetessä ja tuloksen viimeistelyssä. Muulloin yhdistelmäreittien järjestys muodostetaan yhdistämällä vanhat reitit päistään edullisimmalla tavalla ja sekoittamalla syntyneen reitin pinojärjestystä satunnaisesti. Solmujen väliset lyhimmät etäisyydet ja Dijkstran algoritmi

Toistuvien sovitusten menetelmä tarvitsee lähtötiedokseen karttagraafin särmien painot eli puupinojen ja keräilypisteiden väliset pienimmät siirtymäkustannukset ajouria pitkin. Näiden kustannusten oletetaan pysyvän muuttumattomina, joten ne täytyy laskea vain kerran laskennan aluksi. Tämän siirtymäkustannusmatriisin avulla lasketaan toistuvien sovitusten menetelmässä muodostuneiden reittien kokonaiskustannukset.

Niissä optimointitehtävissä, jotka ovat ratkaistavissa kohtuullisessa laskenta-ajassa, voidaan käyttää eksakteja optimointimenetelmiä. Nämä menetelmät antavat tuloksenaan aina aidosti optimaalisen ratkaisun. Esimerkiksi graafin kahden solmukohdan välinen pienin kustannus on P-kompleksisuusluokan ongelma, jonka optimiratkaisu voidaan laskea tarkasti. Tässä työssä käytettiin graafin kahden solmukohdan välisen lyhimmän etäisyyden määrittämiseen Dijkstran algoritmia [12].

Suunnatun tai suuntaamattoman graafin, joilla ei ole negatiivisesti painotettuja sivuja, minkä tahansa kahden solmukohdan välinen lyhin etäisyys voidaan määrittää optimaalisesti Dijkstran algoritmilla. Tämä perustuu kolmioepäyhtälölauseeseen sekä siihen, että lyhimmän kokonaisreitin alireitit ovat myös osaltaan lyhimpiä reittejä. Dijkstran algoritmin rakenne on seuraava:

```
etäisyys(lähtö) ← 0 // Etäisyys lähtösolmuun on nolla.
for all s ∈ Solmut \ {lähtö}
      do etäisyys(s) ← ∞ // Alustetaan muut etäisyydet äärettömiksi.
Käydyt ← ∅  // Alussa käytyjen solmujen joukko on tyhjä.
Käymättömät ← Solmut // Alussa missään solmussa ei olla käyty.
while Käymättömät ≠ ∅ // Tehdään niin kauan kuin on käymättömiä solmuja.
      do u ← minimiEtäisyys(käymättömät,etäisyys)
            Käydyt ← Käydyt ∪ {u} // Lisätään u vierailtuihin solmuihin.
            for all kaikille s ∈ naapurit(u)
                  do if etäisyys(s) > etäisyys(u) + w(u,s)
                        then etäisyys(s) ← etäisyys(u) + w(u,s)
return etäisyys
```

Dijkstran algoritmin tuottaman lyhimmän reitin optimaalisuus todistetaan esimerkiksi lähteessä [30].

4. OPTIMOINTITULOKSEN HYVYYDEN ARVIOINTIMENETELMÄT

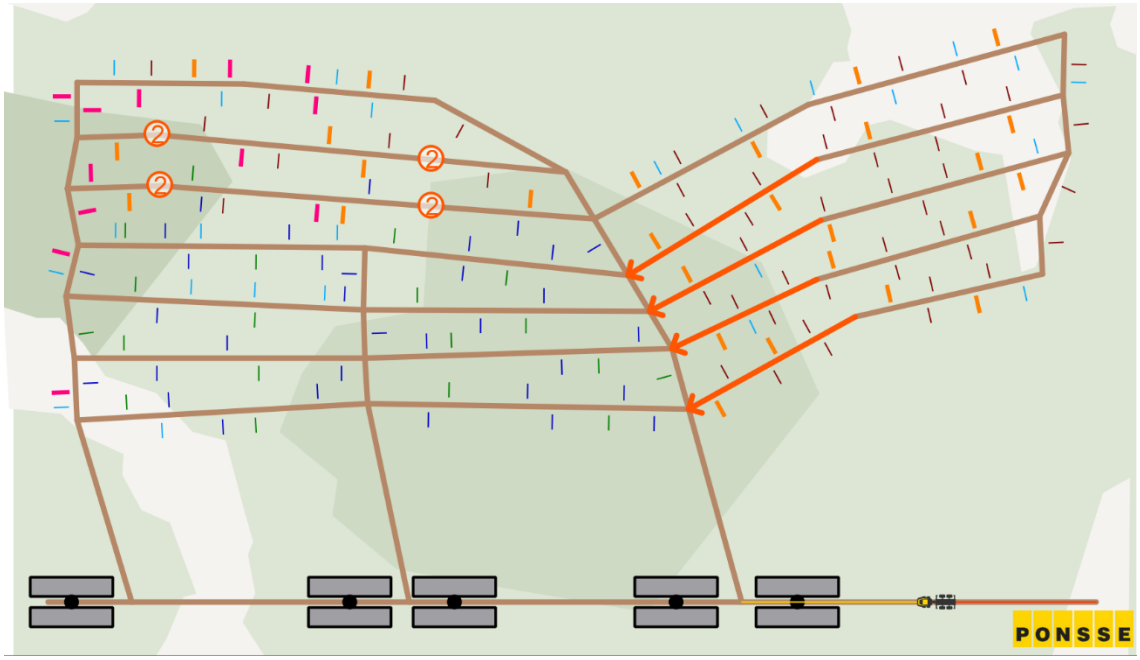
Optimointialgoritmin tehokkuutta tarkasteltiin vertaamalla algoritmin ratkaiseman reitin kokonaiskustannusta metsäkoneenkuljettajien suunnittelemien reittien kokonaiskustannuksiin samassa tehtävässä. Hakkuualuekarttoina käytettiin Ponsse Oy:n omistaman Ponsse Forwarder Game:n pelikarttoja, joilla tulevat ja valmistuneet metsäkoneenkuljettajat harjoittelevat reittisuunnittelua.

4.1 Ponsse Forwarder Game

Ponsse Oy:n omistama Ponsse Forwarder Game (PFG) on tarkoitettu kuormatraktorinkuljettajien koulutukseen. Se tarjoaa Kuva 6 kaltaisen todellista tilannetta simuloivan kartan, jossa kuormatraktori ohjataan hiiri- ja näppäinkomennoilla keräämään kaikki pelileimikolla olevat puupinot alareunassa näkyvän purkusuoran harmaille keräyslavoille. Kuormatraktorin maksimikapasiteetti pelissä on 16 pinoa. Kun kaikki pinot ovat kerättyinä omille puulajikohtaisille lavoilleen, antaa peli yksityiskohtaisen raportin osa- ja kokonaiskeräysajoista.

Metsäkoneenkuljettajien ja optimointialgoritmin suorituskykyä vertaillaan kuvassa 6 nähtävällä Ponsse Forwarder Game-pelissä käytetyllä harvennushakkuualuetta kuvaavalla kartalla. Kyseistä karttaa käytettiin vuonna 2014 ammattikoululaisten Taitaja-kisassa, missä kyseisen vuoden finaaliin selviytyneet metsäkoneenkuljettajaopiskelijat kilpailivat ammattitaidon Suomen mestaruudesta.

Kilpailussa suunniteltujen ja ajettujen reittien kokonaiskeräysaikoja käytetään vertailudatana optimointialgoritmin suorituskykyarvioinnissa.



Kuva 6 *Taitaja-kilpailussa käytetty kartta.* ***[31]***

Kuva 6 Taitaja-kartan ruskeat viivat ovat ajouria, joita pitkin kuormatraktorin tulee kulkea. Alareunan harmaat suorakulmiot ovat lastauspaikkoja eri puulajeille. Ajourien vierellä olevat eriväriset pienemmät suorakulmiot ovat kerättäviä puupinoja. Nämä voidaan kerätä ajouralle pinosta projisoidulta keräyspisteestä. Pinon värikoodi kertoo puulajin. Tässä kartassa kerättävänä on kaikkiaan kuutta eri puulajia. Ajourien päällä olevat numerot ovat ajokertarajoitteita, jotka kertovat kuinka monta kertaa kyseistä uraa voidaan ajaa ilman, että se muuttuu ajokelvottomaksi. Oranssit nuolet puolestaan osoittavat yksisuuntaiset ajouraosuudet.

PFG:n kulkuaika muodostuu kuljetusta matkasta, kuormatraktorin kantamasta kuormasta ja kulkusuunnasta. Kuormatraktoria voi kääntää ainoastaan ajouraristeyksissä, joissa pelin suunnanvaihtonappi aktivoituu. Napin painaminen kääntää traktorin keulan vastakkaiseen ajosuuntaan ja lisää vakiokustannuksen ajoaikaan. Vakiokustannus on suurempi kuin traktorin kääntäminen peruuttamalla risteyksessä siihen tienhaaraan, mihin ei olla menossa ja ajamalla sitten halulle reitille. Tämän vuoksi algoritmin laskemia reittiehdotuksia ajettaessa käytettiin käännyttäessä ainoastaan mainittua peruuttamistapaa.

Yli puolella kuormalla peruuttaminen kasvattaa kulkuaikaa huomattavasti enemmän kuin kustannuskomponenttiensa summan verran. Näin on haluttu korostaa kuinka vältettävää ja vaarallista tällainen ajotapa todellisessa tilanteessa olisi. Kulkuajan lisäksi työaikaa kasvattaa pinojen keruutoimenpiteeseen ja purkutoimenpiteeseen kuluva aika. Peli sisältää myös yksisuuntaiset kulku-urat ja ajokertarajoitetut kulku-urat, kuten kuvasta XX nähdään. Näitä sääntöjä ei pelissä voi rikkoa. Todellisuudessa kuljettaja toki voi yrittää tahallaan tai huomaamattaan rikkoa ohjeistusta, mutta optimointialgoritmin kehittämisen

kannalta tällä ei ole merkitystä. Algoritmi pyrkii antamaan mahdollisimman hyvän keräyssuunnitelman annetuissa rajoitteissa, ja kuljettajan oletetaan toteuttavan tätä suunnitelmaa.

Merkittävää sen sijaan olisi muuttaa reittisuunnitelmaa ajototeutuksen aikana havaittujen reitin ajettavuusmuutosten perusteella. Tätä muuttuvien karttojen toiminnallisuutta ei PFG sisällä, mutta Ponsse Oyj:n edustajien kanssa käytyjen keskustelujen perusteella tämä olisi kuitenkin tulevaisuudessa ensisijainen kehityskohde automatisoidussa reittisuunnittelussa. Tämä ei kuitenkaan vaikuta toistuvien sovitusten menetelmän toimintaan. Reittiosuuksien poissulkeminen muuttaa ainoastaan algoritmille syötettävää graafia, joten toiminnallisuus on tarvittaessa helppo toteuttaa.

PFG katsottiin yllä mainittujen ominaisuuksiensa ja Taitaja-kisassa kerätyn suoritusdatan vuoksi riittävän hyväksi todellisuutta simuloivaksi lähtökohdaksi tässä työssä suunnitellun optimointialgoritmin hyvyyden testaamiseen.

4.2 Optimointitulosten kehittyminen iteraatiokertojen funktiona

Suunniteltua algoritmia testattiin myös itseään vastaan seuraamalla ”Taitaja 2014”-kartan optimointituloksen eli kokonaiskeräysajan muuttumista algoritmin iteraatiokertojen funktiona. Koska kyseessä on NP-vaikea ongelma, ei täysin optimaaliseen ratkaisuun erittäin todennäköisesti koskaan päädytä. Näin ollen iteraatiokertojen määrän lisäämisen vaikutusta optimointituloksen hyvyyteen on perusteltua tarkastella.

Todellisten kuljettajien ajoaikojen perusteella saatiin myös algoritmin haluttu suoritustaso arvioitua. Näin voitiin arvioida mikä on sopiva kompromissi laskenta-ajan ja algoritmin tuloksen hyvyyden välillä. Sopiva laskenta-ajan odotusarvo kerrottiin Ponsse Oy:n edustajien kokemukseen pohjautuvassa vaatimusmäärittelyssä. Heidän mukaansa tilanteita on erilaisia: Toisinaan laskenta voidaan suorittaa yön aikana, jolloin aikaa on useita tunteja. Yleensä kokonaisratkaisun laskenta ei saisi kuitenkaan kestää kuin korkeintaan tunnin siitä, kun harvesterin luoma karttadata on saapunut. Todellisuudessa laskentaa joudutaan suorittamaan muuttuvien olosuhteiden vuoksi myös kuormatraktorin ajosuorituksen aikana, jolloin laskenta-aika saisi kestää vain muutaman minuutin. Luodun algoritmin tehokkuutta tarkasteltiin näissä kolmessa aikaikkunassa

5. OPTIMOINTIALGORITMIN TOTEUTUS

Tässä luvussa kuvataan optimointialgoritmin suunnitteluvalinnat ja yksilöidään kuinka toistuvien sovitusten menetelmää on työssä käytetty sekä perustellaan tehdyt valinnat. Algoritmin rakenteesta käydään läpi luodut funktiot, käytetyt parametrit ja toiminnallisuudet. Erityistä huomiota kiinnitetään valittuihin rajoitteisiin ja karkeistuksiin.

Yksi reitti on tässä työssä kuormatraktorin kulkema matka jostain purku-uran liittymästä määrätyssä järjestyksessä pinolta pinolle lopuksi palaten johonkin purku-uran liittymään. Reittikokoelma sisältää reittejä, jotka täyttävät asetetut rajoitteet ja kokonaisuutena keräävät kaikki pinot annetulta leimikolta vain kerran.

Tarkoitus oli kerätä kaikki pinot kerran ja vain kerran annetuin rajoittein. Kokonaisratkaisu koostuu siis joukosta kapasiteettirajoitettuja reittejä. Jokainen reitti alkaa aina purku-uralta, kulkee hakkuukoneen tekemiä ajouria pitkin yhdeltä kerättävältä pinolta toiselle ja palaa sitten takaisin purku-uralle kuorman purkua varten. Pinoja kerätään metsästä vähintään yksi kappale ja enintään kuormatraktorin kapasiteetin verran. Koska keräysolosuhteiden ei oleteta muuttuvan keräyksen aikana, on reittien keskinäinen järjestys merkityksetön. Kokonaisratkaisun reittien yhteiskustannus on tässä työssä minimoitava suure.

Algoritmi alustetaan lähtöratkaisulla, jossa jokainen reitti sisältää vain yhden kerättävän pinon. Näin ollen alkuperäinen reittisetti sisältää yhtä monta reittiä kuin metsässä on pinoja. Tähän alustusratkaisuun ei oteta ajorajoitettujen urien pinoja, koska niitä ei voida käsitellä tavallisten pinojen lailla ilman, että luodaan runsaasti ajokertarajoitteet ylittäviä ratkaisuja. Tässä työssä ajorajoitteet päätettiin toteuttaa jakamalla kullakin rajoitusuralla olevat pinot yhtä moneen alireittiin, kuin kyseisellä uralla on ajokertoja. Alireittien pinosisältö valittiin muodostamalla mahdollisimman vähän erilaisia puulajeja sisältäviä alireittejä. Nämä alireitit lisättiin sitten jokaisella algoritmin iteraatiokierroksella toistuvien sovitusten menetelmän tuottamaan ratkaisuun. Lisäys suoritettiin kokeilemalla kutakin alireittiä jokaiseen mahdolliseen liitoskohtaan ja valitsemalla pienimmän kustannuksen tuottava liitoskohta.

Alustettu reittisetti toimii yhdessä kustannusmatriisin kanssa syötteenä toistuvien sovitusten algoritmille, joka pyrkii yhdistelemään näistä reiteistä kustannukseltaan halvempia reittejä. Lähtöratkaisu on täydellinen sovitus ja jokainen edellisestä ratkaisusta muodostettava seuraava ratkaisu säilyttää myös täydellisen sovituksen vaatimuksen. Näin yksikään pino ei jää keräämättä tai tule kerätyksi kahdesti.

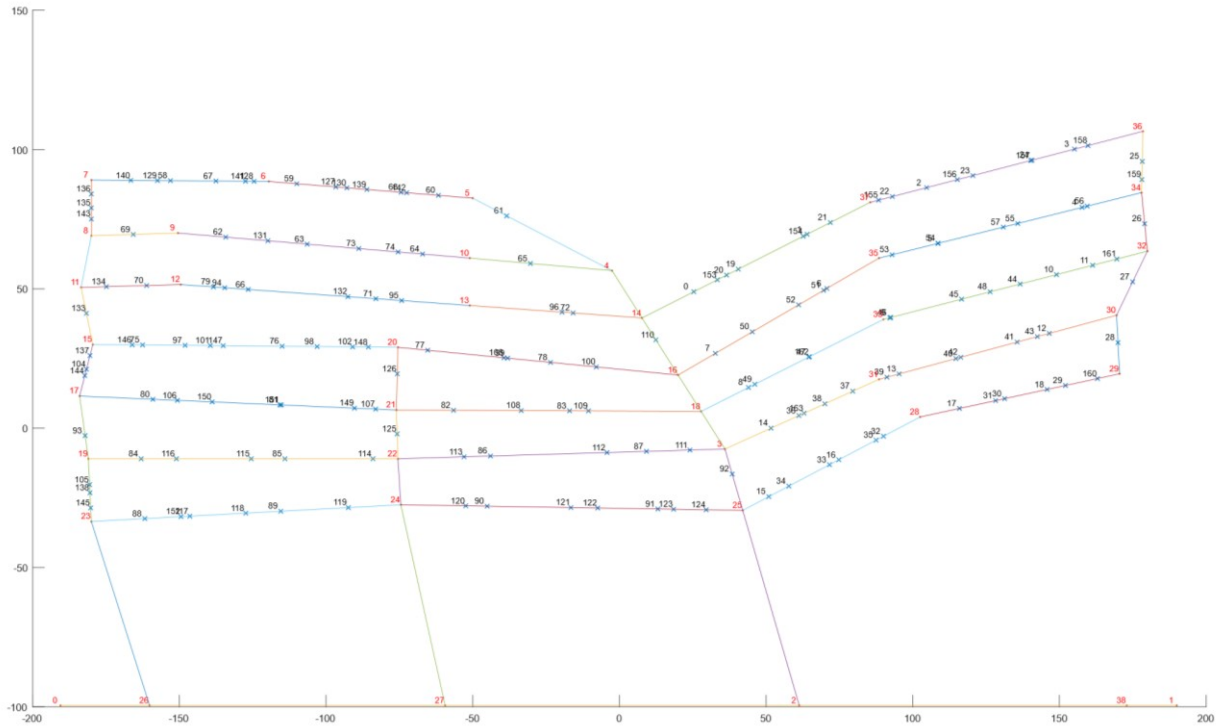
Lopuksi ratkaisuun yhdistetään vielä ajorajoiteurien sisältämien reittien pinot. Ajorajoiteurien reitit ratkaistaan erillisesti muusta optimoinnista. Kun tämä kokonaisratkaisu on

valmis, määritetään purkukustannukset minimoiva puulajikonfiguraatio purku-uran keruulavoille. Tämän jälkeen ratkaisu sisältää kaikki tarvittavat tiedot, jotka kuormatraktorin kuljettaja tarvitsee keruutehtävän suorittamiseen.

5.1 Algoritmin karttatiedot

Tehtävä alkaa karttadatan muokkaamisesta algoritmin olettamaan muotoiluun. Ensin kerätään tiedot reittien solmukohtien eli ajouraristeysten ja pinojen koordinaateista sekä kuljettavista särmistä näiden välillä. Näistä luodaan vieruspistematriisi, jossa elementteinä ovat siirtymäkustannukset kahden vieruspisteen välillä. Taitaja-kartassa pisteiden välinen kustannus on aina lyhin etäisyys kyseisten pisteiden väillä. Todellisuudessa kustannus muodostuu GPS-datan perusteella muodostetusta uran pituudesta ja mahdollisista muista siirtymäkustannuksia mallintavista kustannuskomponenteista. Koska osa reiteistä on yksisuuntaisia, on karttagraafi suunnattu, eikä vieruspistematriisi ole symmetrinen.

Kuva 7 nähdään ”Taitaja 2014” -kartan XML-datasta Matlab-ohjelmistoon siirretyn karttagraafin tuloste. Punaiset pisteet ovat risteyksiä ja mustat pisteet ovat pinojen poimintakohtia. Numerot ovat pinojen ja risteysten tunnisteita, joita vastaavat sijainti- ja puulajitiedot ovat tallennettuina näitä vastaaviin matriiseihin.



Kuva 7. *Taitaja-kartan graafi piirrettynä kaksiulotteisessa koordinaatistossa Matlab-ohjelmistolla.*

Näiden pohjatietojen avulla voidaan muodostaa vieruspistematriisi, jonka pohjalta Djikstran algoritmilla määritetään tarvittavat pinojen väliset pienimmät siirtymäkustannukset.

5.2 Siirtymäkustannusmatriisin muodostaminen vieruspistematriisista

Pohjatiedokseen algoritmi tarvitsee kustannusmatriisin, joka sisältää pienimmät siirtymäkustannukset kaikkien leimikolta kerättäviksi tarkoitettujen pinojen välillä. Harvesterin GPS:n perusteella saatu karttatieto pinojen ja ajourien sijainnista muunnetaan vieruspistematriisiksi. Se kuvastaa graafin solmukohtien l. hakkuualueelle tehtyjen puupinojen välisiä olemassa olevia yhteyksiä näiden välisillä etäisyyksillä. Tässä vaiheessa pisteiden välimatkaan perustuva kustannusrakenne voidaan myös korvata millä tahansa muulla todellisia siirtymäkustannuksia kuvaavalla kustannusrakenteella, kuten kulkuajalla, energiakustannuksella tai näiden yhdistelmällä. Tärkeintä on, että saadaan muodostettua graafi, joka kuvastaa pisteiden välisiä siirtymäkustannuksia mahdollisimman tarkkaan käytännön päätöskriteerien kannalta.

Kuva 8 havainnollistetaan, kuinka graafista muodostetaan vieruspistematriisi. Kun suunnatun graafin G solmukohdasta i on suora yhteys solmukohtaan j, merkitään vastaavaan vieruspistematriisin N elementti N_{ij} nollasta poikkeavaksi. Usein painotetun graafin tapauksessa luvuksi merkitään kyseisen sivun paino, jolloin vältetään tarve erilliselle kustannusmatriisille. Kun suoraa yhteyttä ei ole, merkitään vastaavaan elementtiin arvoksi 0. Tässä vaiheessa saadaan algoritmille näin helposti määrättyä kartan kielletyt ajosuunnat.

Kuva 8. *Suunnatun graafin esittäminen vieruspistematriisina.*

Vieruspistematriisin avulla voidaan laskea Dijkstran algoritmilla lyhimmät etäisyydet pinojen väleille. Tämä ei vaikuta laskenta-aikaan merkittävästi, sillä kustannusmatriisi määritetään vain kerran, toisin kuin sovituskustannusmatriisi, joka muuttuu dynaamisesti jokaisella iteraatiokierroksella. Dijkstran algoritmilla laskettuihin lyhimpiin reitteihin liittyvät polut ja kustannukset tallennetaan polkumatriisiin ja siirtymäkustannusmatriisiin.

Polkumatriisia käytetään sekä ratkaistujen reittien piirtämiseksi kuljettajalle, että algoritmin ajon aikana peruutuskustannusten määrittämiseen. Polkumatriisin polut sisältävät myös käytetyt ajouraristeykset oikeassa järjestyksessä. Toisin sanoen, jos pinolle tullaan samalta suunnalta kuin mihin lähdetään, täytyy ajoneuvon kulkea peruuttamalla. Vain ajouraristeyksissä voidaan kääntyä ja missään näistä risteyksistä ei ole kerättäviä pinoja. Seuraavassa ajouraristeyksessä lisätään reittikustannukseen kääntymiskustannus ja lopetetaan peruutuskustannuksen kasvattaminen.

5.3 Sovituskustannusmatriisin muodostaminen

Sovituskustannusmatriisi muodostetaan edellisen iteraation ratkaisun reittien pohjalta. Tarkoituksena on muodostaa uusia reittejä yhdistämällä kaksi reittiä uudeksi reitiksi tai vaihtamalla pinoja kahden reitin välillä. Yhdistetty reitti luodaan, jos pinoja on yhteensä vähemmän kuin maksimikapasiteetti, ottamalla molempien reittien sisältämät pinot ja yhdistämällä ne yhdelle reitille, joko kokonaislukuoptimoinnilla tai reittien päistä mahdollisimman pienellä kustannuksella. Laskennan alussa käytetään nopeampaa reittien päistä tapahtuvaa yhdistämistä, koska reittien pinosisältöjen halutaan lähestyvän maksimikapasiteettia, ennen kuin kauppamatkustajan ongelman ratkaisua käytetään reitin pinojärjestyksen optimoimiseksi. Tämän jälkeen optimaalista järjestelyä käytetään sadan iteraation välein ja aina ratkaisun viimeistelyyn algoritmin lopetusehdon täytyttyä. Reittien päillä tarkoitetaan tässä tapauksessa ratkaistujen reittien ensimmäisiä ja viimeisiä pinoja. Eli purku-urien liittymätiedot poistetaan yhdistettävistä reiteistä ja ne liitetään uudestaan uuden reitin päihin.

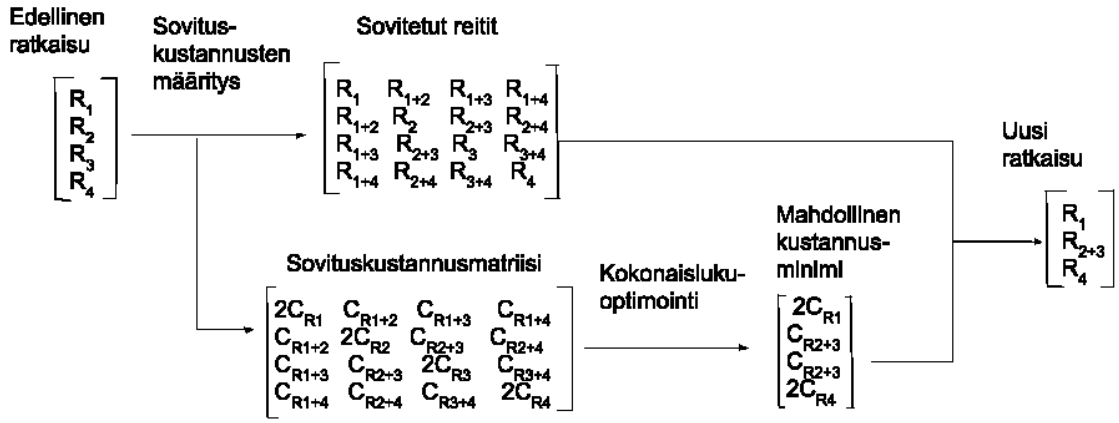
Toinen paritustapa vaihtaa satunnaisesti yhdestä kolmeen pinoa kahden reitin välillä. Valinnan todennäköisyysjakaumana käytettiin tasajakaumaa. Tämä vaihto toistetaan niin kauan, kuin uusien reittien yhteiskustannus pienenee, jos kustannus ei pienene, säilytetään aikaisempi ratkaisu. Huomattakoon, että tämä paritustapa yrittää järjestellä maksimikapasiteetinkin käyttäviä reittejä uusin tavoin.

Näitä kahta sovituskustannusta verrataan toisiinsa sekä alkuperäiseen kustannukseen, minkä jälkeen valitaan halvin tapa reittien sovituskustannukseksi tällä iteraatiokierroksella. Matriisin diagonaalille lasketaan reittien kustannukset itsensä kanssa. Tämä itsesovituskustannus on suoraan kyseisen reitin reittikustannus kaksinkertaisena. Kaksinkertaisuus on tarpeellista kokonaislukuoptimoinnin ratkaisutavan vuoksi. Diagonaalielementit ovat matriisissa vain kerran, kun taas alakolmiomatriisiin lasketut parituskustannukset peilataan yläkolmiomatriisiin siten, että kustannukset $C_{ij} = C_{ji}$.

Nyt kun kaikkien reittikokoelman reittien kaikki mahdolliset sovituskustannukset on määritetty, voidaan syntynyt sovituskustannusmatriisi lähettää syötteeksi toistuvien sovitusten menetelmän kokonaislukuoptimointiin.

Kuvassa 9 havainnollistetaan kuinka sovituskustannusmatriisi rakentuisi neljän reitin ratkaisusta ja kuinka sitä käytetään toistuvien sovitusten menetelmässä. Sovituskustan-

nusten määritys tuottaa lähtöratkaisun pohjalta kaksi toisiinsa liittyvää matriisia. Ensimmäinen matriisi kertoo yhdistetyt reitit ja toinen näiden kustannukset vastaavissa matriisien elementeissä. Sovituskustannusmatriisia käytetään herätteenä kokonaislukuoptimoinnissa, joka esimerkissä valitsee neljän reitin tapauksessa sovituskustannusmatriisista ratkaisuksi aina neljän elementin yhdistelmän.

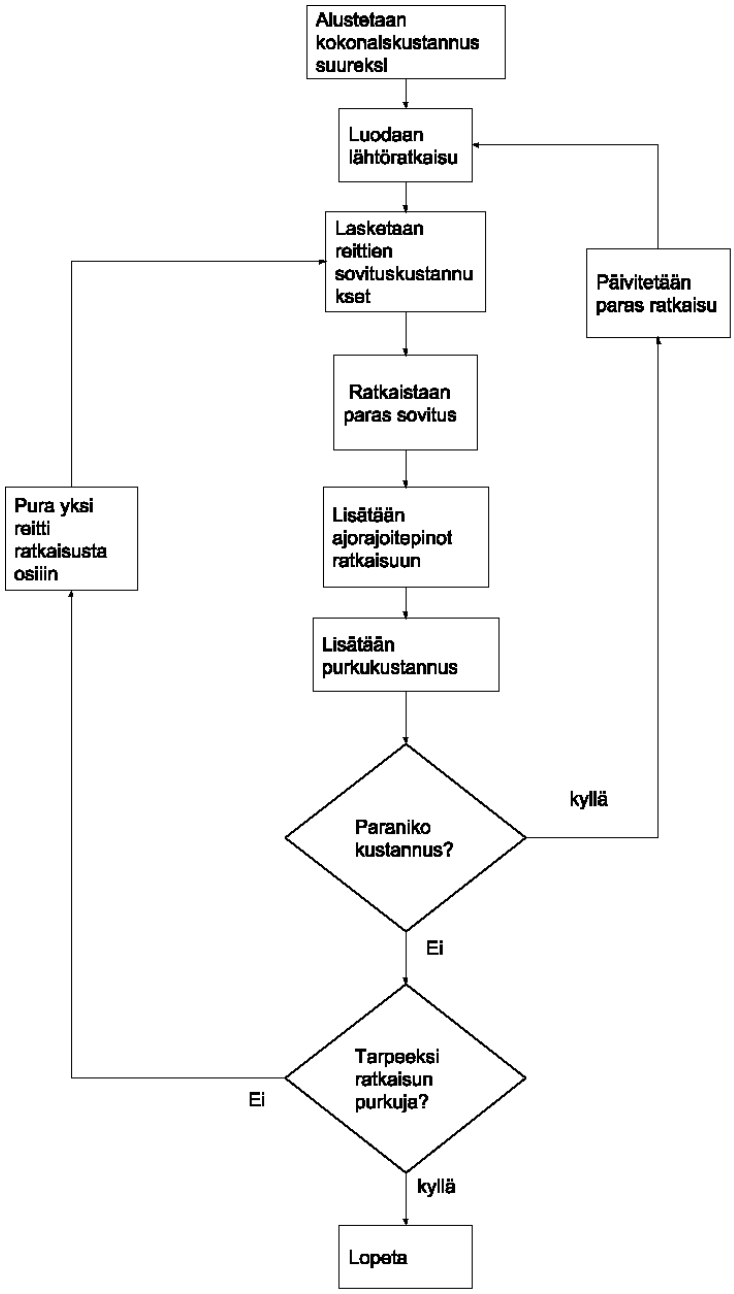


Kuva 9. *Sovituskustannusmatriisin rooli toistuvien sovitusten menetelmässä.*

Kuvan esimerkissä reitit 1 ja 4 ovat pysyneet ennallaan ja reitit 2 ja 3 ovat yhdistettyjä. Sovituskustannusmatriisi ei sisällä informaatiota reittien yhdistämistavoista, joten nämä haetaan ratkaisuun reittimatriisista. Sen avulla tarkastellaan, onko reitti R_{2+3} yhdistelmäreitti vai kaksi uudelleenjärjesteltyä reittiä. Jälkimmäisessä tapauksessa ratkaisuun tulee neljä reittiä, joista kaksi on pysynyt samana kuin lähtöratkaisussa ja kaksi muuttunut pinosisällöltään tai -järjestykseltään. Edellisessä tapauksessa ratkaisun reittimäärä pienenee kolmeen.

5.4 Toistuvien sovitusten menetelmän toteutus

Edellisessä vaiheessa muodostettu sovituskustannusmatriisi on perustana tälle vaiheelle. Sovituskustannukset lasketaan uudestaan jokaisella algoritmin iteraatiokierroksella. Tämä toimii yhtenä syötteenä Matlabin kokonaislukuoptimointiratkaisijalle. Ratkaisijaetsii halvimman kokonaiskustannuksen tuottavan yhdistelmän sovituskustannusmatriisin reiteistä. Toistuvien sovitusten menetelmän havainnollistava vuokaavio nähdään Kuva 10.

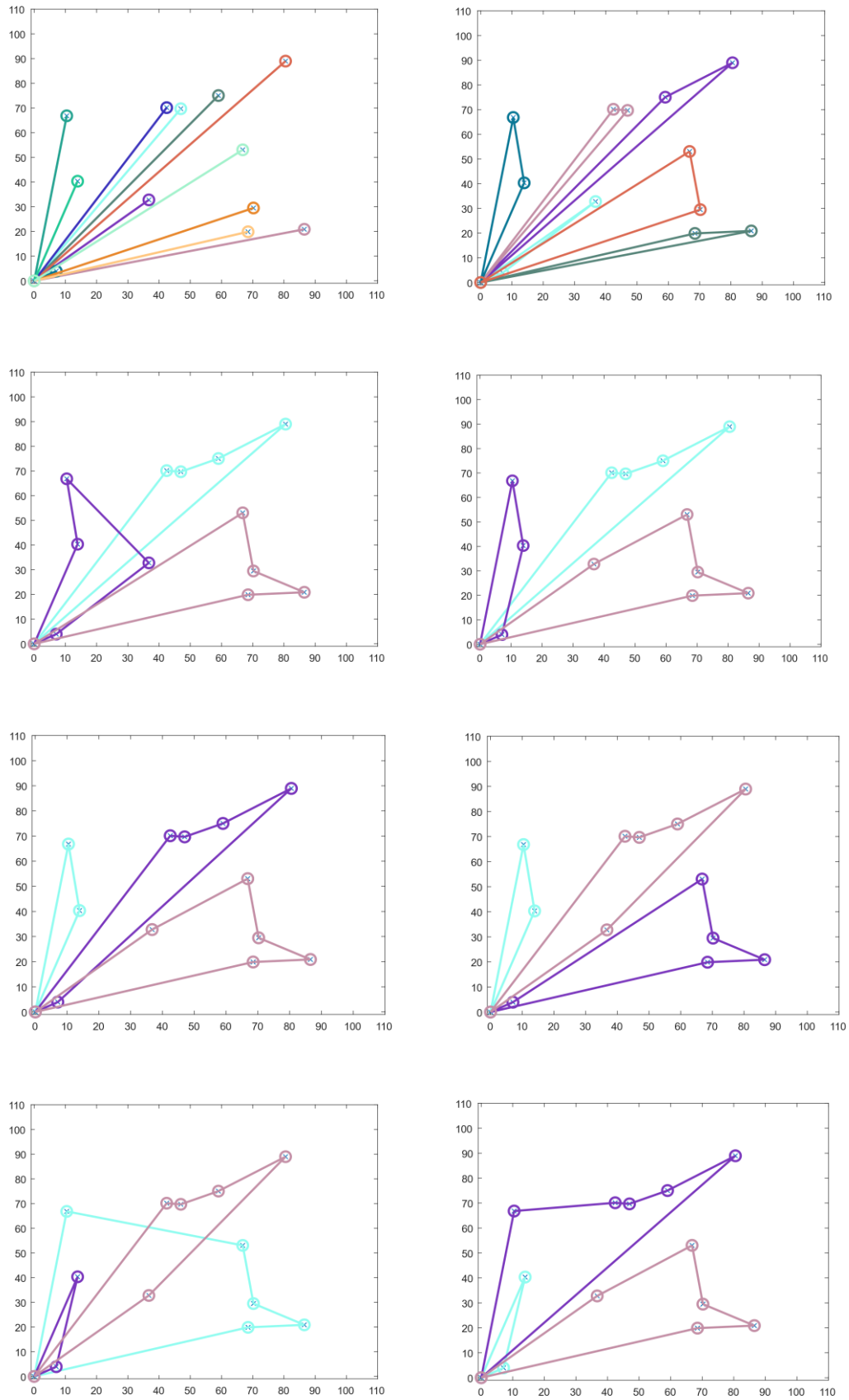


Kuva 10. *Toistuvien sovitusten menetelmän algoritmi*

Kokonaisratkaisussa täytyy olla jokainen pino kerättynä vain kertaalleen. Tämä varmistetaan ratkaisijan epäyhtälörajoitteilla ja yhtälörajoitteilla. Vastauksena saadaan binääriarvoinen vektori, joka määrittää, mikä edellisen ratkaisun reittien täydellinen sovitus tuottaa uuteen ratkaisuun pienimmän kokonaiskustannuksen. Tätä ratkaisua verrataan aikaisempaan parhaaseen ratkaisuun ja parempi ratkaisu säilytetään. Kyseistä rakennetta toistamalla päästään yhä parempiin ratkaisuihin, kunnes lopulta jäädään johonkin ääriarvoon.

Kuva 11 nähdään, kuinka ratkaistu reittikokoelma kehittyy kokonaiskustannuksen parantuessa, kun pinoja on leimikolla 12 ja ajoneuvon kapasiteetti on 5 pinoa. Pinojen väliset kustannukset ovat havainnollisuuden vuoksi nyt suoraan niiden väliset euklidiset etäisyydet. Aluksi reitit pitenevät maksimipituuksiinsa, jonka jälkeen pinoja reittien välillä vaihtelemalla haetaan yhä parempia reittejä ratkaisuun. Lopulta ratkaisu ei enää parane ja näin saatu optimiratkaisu tallennetaan.

Koska algoritmilla on vaarana loukkuuntua paikalliseen optimiin, ratkaisua muokataan hajottamalla muutama kokonaisratkaisun reitti osiin siten, että ratkaisu muuttuu merkittävästi ja loukkuuntumisen vaara vähenee. Ratkaisun hajottamisella tarkoitetaan yksinkertaisesti sitä, että joku ratkaisun reiteistä muutetaan reittijoukoksi, jossa reitin pinot käydään hakemassa yksitellen, lähtien ja palaten lähimpään purku-uraliittymään. Näin ollen esimerkiksi ratkaisun yksi kahdeksan pinon reitti hajotettaisiin kahdeksaksi yhden pinon reitiksi.



Kuva 11. *Ratkaisun kehittyminen toistuvien sovitusten menetelmällä. Alustusratkaisu on vasemmalla ylhäällä ja paras ratkaisu oikealla alhaalla.*

Hajotettu ratkaisu syötetään taas toistuvien sovitusten menetelmään ja algoritmin mainitut sisäiset satunnaisuudet reittien parituksessa ajavat tuloksen mahdollisesti johonkin toiseen paikalliseen minimiin. Näin ratkaisu kehittyy iteratiivisesti paremmaksi tai pysyy vähintään yhtä hyvänä. Ohjelma 1 kuvaa käytetyn menetelmän pseudokoodina.

Ohjelma 1. *Toistuvien sovitusten menetelmän pseudokoodi*

```
algorithm Toistuvien sovitusten menetelmä(kustannukset, polut, kapasiteetti,
pinoja, ajorajoitereitit)

   set parasKustannus = inf

   set hajoituskertojenEnimmäismäärä, enimmäismäärä

   set ratkaisu = alustettuRatkaisu(pinoja,kapasiteetti)

      while ratkaisunHajoitusKerrat < hajoituskertojenEnimmäismäärä

         while parasKustannusEiParantunut < enimmäismäärä

            sovituskustannusMatriisi = parituskustannukset (ratkaisu, kustan-
            nukset, polut, kapasiteetti)

            yhtälörajoitteet = määritäYhtälörajoitteet (sovituskustannusMat-
            riisi)

            uusiRatkaisu = kokonaislukuoptimointi (yhtälörajoitteet, sovitus-
            kustannusMatriisi)

            uusiKokonaisRatkaisu = lisääAjorajoitepinot (ajorajoitereitit, uu-
            siRatkaisu)

            purkuLavat = luoLavakonfiguraatio(uusiKokonaisratkaisu)

            purkuKustannus = määritäPurkukustannus(purkulavat,uusiKokonaisrat-
            kaisu)

            uusiKustannus = laskeKustannus(uusiKokonaisratkaisu,purkuKustan-
            nus)

            if uusiKustannus < parasKustannus

               parasKustannus = uusiKustannus

               ratkaisu = uusiRatkaisu

            else

               kasvata parasKustannusEiParantunut

            end if

         end while

         ratkaisu = hajoitaRatkaisu(ratkaisu)

               kasvata ratkaisunHajoitusKerrat
```

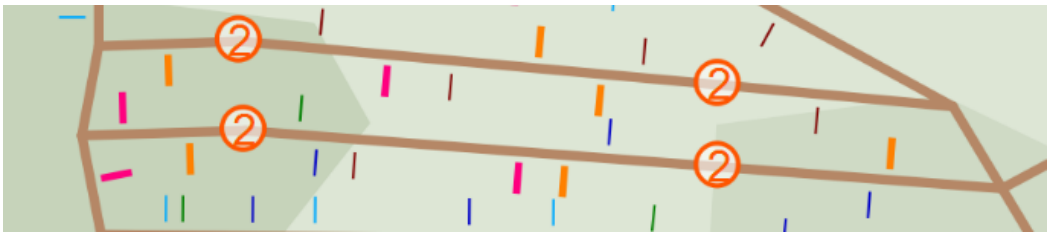
```
    end while
end algorithm
```

Näin saatuun ratkaisuun yhdistetään vielä pääoptimoinnista erossa pidetyt ajorajoiteurapinot. Lopuksi ratkaisulle määritetään mahdollisimman hyvä purkulavakonfiguraatio, eli mitä puulajia kannattaa millekin keruulavalle purkaa. Nämä käydään läpi seuraavaksi.

5.5 Ajourien ajokertarajoitteiden toteutus

Ajokertarajoitetut ajourat, ovat ne kartan ajourat, jotka kestävät vain määrätyn määrän kulkukertoja ennen ajokelvottomaksi muuttumistaan. Ajosuunnalla ei ole tämän rajoitteen kannalta merkitystä. Kuva 12 nähdään PFG:n ajokertarajoitetoteutus. Siinä ajokertarajoitteet on toteutettu rajoitetun ajourasegmentin päätepisteiden maksimivierailukertoina. Käytännössä tämä varmistaa, ettei ajouralla käydä yhteensä päätepisteissä oleviin ympyröihin merkittyä kahta kertaa enempää.

Ajokertarajoitteiden sisällyttämiseksi optimointiin pohdittiin kahta eri tapaa. Ensimmäisessä lisättiin ajokertarajoitteet ylittäviin ratkaisuihin liki ääretön kustannussakko, jolloin nämä ratkaisut häviäisivät sallitun määrän rajoiteuria käyttäneille kokonaisratkaisuille. Koska kuitenkin suurin osa syntyvistä ratkaisuista on kelvottomia, kun ajorajoitetulla uralla on enemmän eri puulajeja kuin sallittuja ajokertoja, tekee tämä laskennan suppenemisesta hidasta. Kuva 12 kartalla alemmalla ajokertarajoitetulla uralla on jopa viittä eri puulajia. Tämä pakottaa keräämään yhden vähintään kolmea puulajia sisältävän kuorman, jonka purkaminen ei näin ollen onnistu yhdestä purkupisteestä. Taitaja-kartalla yhdestä purku-uran purkupisteestä voi purkaa korkeintaan kahta eri puulajia.



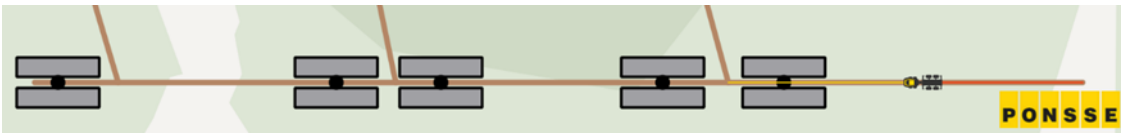
Kuva 12. *Taitaja-kartan ajokertarajoitetut urat.* ***[31]***

Toinen tapa, jota päädyttiin käyttämään lopullisessa algoritmissa, oli muodostaa ajorajoiteurien pinoista mahdollisimman vähän puulajivaihtoja sisältäviä alireittejä, jotka sitten lisättiin optimaalisesti toistuvien sovitusten menetelmän ratkaisuun, eli ratkaisuun, joka sisältää keräysreitit muiden kuin ajokertarajoitepinojen osalta. Alireittejä kokeiltiin molemmin päin kaikkiin mahdollisiin ratkaistujen reittien pinoväleihin, sekä reittien alkuihin ja loppuihin. Pienimmän kustannuksen tuottava yhdistelmä valittiin tämän vaiheen jälkeiseksi kokonaisratkaisuksi.

Tässä tapauksessa, kun alireittejä syntyi vain neljä kappaletta eikä pinoja leimikolla ole kuin 164 kappaletta, oli toinen tapa ensimmäistä tehokkaampi laskenta-ajassa mitattuna. Suuremmilla leimikoilla, joissa on enemmän pinoja, ajokertarajoitettuja uria ja puulajeja, voidaan alireitit yhdistää satunnaisesti tai jollain sopivaksi katsotulla metaheuristiikalla muiden pinojen kokonaisratkaisuun. Tällöin täytyy kuitenkin ottaa huomioon, että aiempi ratkaisu saattaa hylätä heikompina ratkaisuja, jotka ajokertarajoitettujen alireittien lisäämisen kannalta olisivat kuitenkin parempia ratkaisuja. Toisin sanoen alkuperäisen osaratkaisun ja yhdistetyn ratkaisun paikalliset ääriarvokohdat todennäköisesti poikkeavat toisistaan. Alkuperäisen osaratkaisun, eli toistuvien sovitusten menetelmän ratkaisun, sisältämä satunnaisuus kuitenkin lieventää tätä ongelmaan, koska toisinaan ratkaisu voi lähteä yhdistelmäratkaisulle suotuisampaan suuntaan.

5.6 Kuorman purkuoptimoinnin toteutus

Käytännössä purku-uraoptimoinnin päämäärä on valita tehokkain keruulavojen puulajikonfiguraatio, eli kuinka lavojen puulajit täytyy valita, jotta pääoptimointitehtävässä saatujen kuormien purku käy pienimmin kustannuksin. Taitaja-kartan purku-ura nähdään Kuva 13. Kukin harmaista lavoista voi sisältää vain yhtä puulajia, jonka kuljettaja voi PFG:ssä itse määrittää purkuvaiheessa. Kuvassa näkyy myös kolme risteystä, joista kustakin pääsee eri kohtiin samaa leimikkoa.



Kuva 13. *Taitaja-kartan purku-ura.* ***[31]***

Aluksi kokeiltiin laskea sopivin lavakonfiguraatio pääoptimointitulokselle, joka sisälsi purkuliittymiin tulevat kuormat. Purkuliittymät kuormille valittiin viimeisiksi kerättyjen pinojen läheisimmiksi liittymiksi. Tällöin pääratkaisu on kuitenkin muodostunut täysin purkutehtävästä riippumattomasti, jolloin usein mikään lavakonfiguraatio ei ollut riittävän hyvä ja purkamiseen käytetty aika muodostui suureksi. Tämän vuoksi lopulta päädyttiin sisällyttämään myös purkulavakonfiguraation muodostaminen pääoptimointifunktion viimeiseksi vaiheeksi.

Toistuvien sovitusten menetelmän ja siihen lisätyn ajorajoiteuraratkaisun perusteella tiedetään, mitä kuormia tulee mihinkin purku-uraliittymään. Koska kaikkien lavakombinaatioiden laskenta on myös NP-vaikean kompleksisuusluokan ongelma, johon täytyy pystyä saamaan nopea ratkaisu jokaiselle pääoptimointi-iteraatiolle, muodostetaan lava-puulajiparit satunnaisesti tasajakaumalla. Tämän jälkeen lasketaan reittien muodostamien kuormien purkukustannus arvotulla lavajärjestyksellä. Kyseisen purkulavasilmukan toistokertojen määrä yhdessä pääoptimointikierroksessa toimii samalla kokonaisoptimoinnin viritysparametrinä. Lopullinen kokonaisratkaisu muodostetaan yhdistämällä

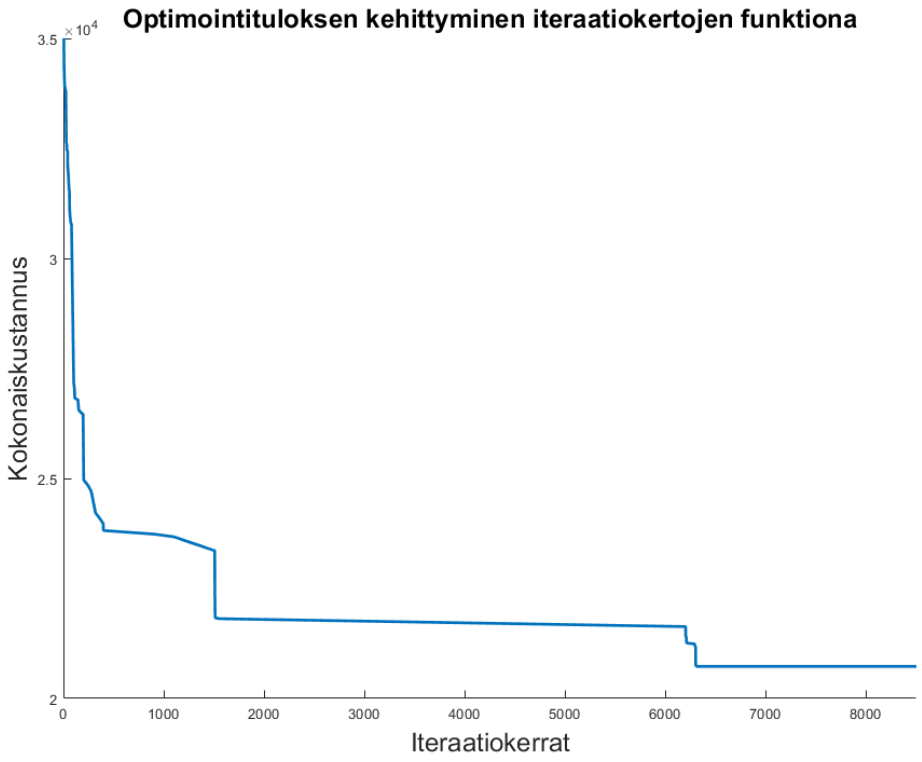
valitun purkulavakonfiguraation tuottama purkukustannus pääoptimoinnissa saatuun pinojen keräyskustannukseen, johon on lisätty jo ajokertarajoitetut osareitit.

Tässä vaiheessa oli selvästi havaittavissa, että eri puulajien määrä yhdessä kuormassa olisi suhteutettava purku-uran lavasijaintien mukaan. Taitaja-kartalla jokaisesta purkupisteestä pystyy purkamaan kahdelle eri lavalle, joten tämä määrä asetettiin tavoitearvoksi kuorman puulajimäärälle asettamalla ylimääräinen kustannus jokaiselle kahden puulajin ylittämälle kuormalle. Huomattiin myös, että purkulavojen alustuskonfiguraatio kannattaa muodostaa kunkin puulajin kerättävien pinojen massakeskipisteiden suhteella. Yksinkertaistaen, jos leimikolla olevan puulajin odotusarvoinen sijainti on leimikon vasemmassa reunassa, on sen keräyslavankin hyvä olla vastaavasti purku-uran vasemmassa reunassa. Kun päälaskenta on suoritettu, lasketaan purkulavakonfiguraatio vielä kerran ratkaisun tuottamalle reittikokoelmalle huomattavasti suuremmalla iteraatiomäärällä.

6. TULOKSET

Suunnitellun optimointialgoritmin tärkeimmät laskenta-aikaan ja tuloksen hyvyyteen vaikuttavat parametrit olivat algoritmin ratkaisun hajotusten vähimmäismäärä, ratkaisun hajotukseen johtava perättäisten hyödyttömien iteraatiokierrosten määrä jolloin ratkaisu ei ole parantunut. Näiden lisäksi tuloksen hyvyyteen vaikutti erityisesti yksittäisten reittien kustannuksen määräävän funktion parametrit: peruutuskustannus ja yhden kuorman sisältämän erilaisten puulajien määrän mukana kasvava kustannus. Peruutuskustannus laskettiin algoritmissa mukaan vain, kun kuorman kapasiteetista oli käytössä yli puolet. Tämä on voimakas karkeistus todellisuudesta, jossa jokainen puu kuormassa luonnollisesti lisää liikkumisesta aiheutuvaa kustannusta. Laskenta-aikaa voitiin myös rajoittaa iteraatioiden maksimäärällä, mutta tämä pidettiin nyt vain kyllin korkeana, jotta käytännössä laskennan pysäyttäisi vaadittu ratkaisun hajotusmäärän täyttyminen.

Tuloksen kehittymistä tarkasteltiin ajamalla kehitettyä algoritmia Taitaja-kartan graafilla mahdollisimman pitkään. Yksi kuvaavaksi katsottu esimerkki tällaisen ajon tuloksen kehittymisestä iteraatiokertojen funktiona nähdään Kuva 14. Siinä nähdään n. tunnin mittainen alkuosa vuorokauden mittaisesta ajosta, jonka lopullinen kustannus oli $1.8782 \cdot 10^4$ yksikköä. Ratkaisu oli parantunut viimeisen kerran noin 8 tunnin ajon jälkeen. Vaatimusmäärittelyssä algoritmin laskenta-ajaksi toivottiin korkeintaan tuntia. Tunnin kohdalla käytetty laitteisto oli ehtinyt kuvan esimerkissä laskea n. 8000 iteraatiota, jolloin kustannus oli $2.0729 \cdot 10^4$ yksikköä, eli n 10%:n päässä vuorokauden laskennan tuloksesta.



Kuva 14. *Optimointituloksen kehittyminen iteraatiokertojen funktiona.*

Taulukossa 1 ovat ”Taitaja 2014”-kilpailun finalistien tulokset, joita käytettiin vertailudatana. Suunnitellun algoritmin tulokset ovat alimpana taulukossa tunnuksella RM. Kirjatut tulokset ovat algoritmin parhaimmat tulokset tunnin ja minuutin laskenta-ajoilla normaalitehoisella kotitietokoneella. Vuorokauden laskenta-ajoilla tuotos ei merkittävästi muuttunut tunnin laskenta-ajoista. Tässä vaiheessa algoritmin kustannusrakenteen poikkeavuudet PFG:n kustannusrakenteen kanssa vaikuttivat enemmän tuottoihin kuin laskenta-aika. Algoritmin muodostamaa reittikokonaisuutta PFG:ssä ajettaessa, oli havaittavissa, ettei algoritmin muodostama kustannusarvio vastannut aina PFG:ssä toteutunutta tuotosta.

Tämä on seurausta mallinnusvirheestä algoritmin kustannusmatriisin ja pelin todellisten kustannusten muodostumisen välillä. Algoritmi olettaa esimerkiksi kuorman toteuttavan LIFO-järjestelmää, mutta PFG:ssä kuorma rakentuukin kerroksittain ensin lavan reunoille ja sitten keskelle vasemmalta oikealle. Purettaessa keskelle kuormaa jääneet puulajit aiheuttavat pahimmillaan satojen metrien ylimääräisiä edestakaisia liikkeitä purkuuralla.

***Taulukko 1**. Taitaja-kilpailun finalistien suoritukset verrattuna suunniteltuun algoritmiin*

Kuljettaja	Ajon vaiheet					Ajettu matka / (km)	Kuormien lukumäärä	Tuotos / (m3/h)
	Tyhjänä ajaminen / (min)	Kuormausajo / (min)	Täytenä ajaminen / (min)	Purkaminen / (min)	Kokonaisaika/ (min)			
Taitaja 2	40	213	31	68	352	9.2	11	27.9
Taitaja 9	62	187	47	79	375	11.1	12	26.1
Taitaja 3	66	213	40	61	380	11.2	13	26
Taitaja 1	58.5	227	39.5	75.5	400.5	12.3	12	24.5
Taitaja 7	54	232	32.5	81	399.5	12.2	11	24.5
Taitaja 8	53	232	48	84	417	12.8	11	23.6
Taitaja 5	45	260	47	66	418	11.1	11	23.2
Taitaja 6	65	214	58	98	435	12.8	13	22.6
Taitaja 4	48.5	245	38	110	441	14.4	11	22.2
RM (h)	40	271	33	75	419	12.8	12	23.4
RM (min)	44	343	32	84	503	16.5	12	19.5

Myös peruutuskustannukset ja kuorman massan aiheuttamat siirtymisten lisäkustannukset poikkeavat algoritmin ja PFG:n välillä jonkin verran. Nämä voidaan kuitenkin helposti tarvittaessa korjata todellisessa tilanteessa korjaamalla algoritmin kustannusmallia puunkorjuusta saadun datan perusteella.

Taulukosta xx. voidaan nähdä, että algoritmi toimii Taitaja-kilpailun kartalla hiukan finalistien keskitason alapuolella vaatimusmäärittelyssä toivotulla tunnin laskenta-ajalla. On kuitenkin huomioitava, että yhtenevämmällä kustannusrakenteella algoritmin ja PFG:n välillä algoritmin saavuttamaa tuotosta voidaan vielä parantaa. Laskentanopeutta puolestaan voidaan jonkun verran parantaa säätämällä ratkaisun purkuväliä, purkufunktion toteutusta ja alustavan ratkaisun rakennetta.

Verrattaessa ajon eri vaiheita ihmisten ja algoritmin välillä on nähtävissä, että algoritmi käyttää vain vähän aikaa tyhjänä ja täytenä ajamiseen, mutta kuormausajoreitit ovat hitaampia kuin kenelläkään ihmiskuljettajista. Kun katsotaan loppuraportista, mitkä reiteistä johtivat huonoimpiin tuotoksiin, nähdään, että ajorajoitettuja uria sisältäneet reitit olivat tuotokseltaan keskimäärin noin 4 m^3/h huonompia, kuin muut reitit. Tämä on merkittävä määrä, kun huomioidaan, että algoritmin kokonaisratkaisussa on yleensä neljä reittiä, jotka kulkevat ajorajoitettujen urien kautta.

Algoritmin tuleviin versioihin onkin syytä muuttaa ajorajoitepinojen yhdistäminen jo toistuvien sovitusten menetelmän sisälle. Sen sijaan purkuoptimointi vaikuttaisi tulosten valossa olevan kilpailukykyinen tällaisenaankin, vaikka kuorman koostumuksesta johtu-

neet edestakaiset ajot aiheuttivatkin turhaa edestakaista ajoa purku-uralla. Koska purkuoptimointi ei kuitenkaan ole täysin riippumaton pääoptimoinnista, ei voida tietää kuinka paljon todellisuudessa parempia kokonaisratkaisuja algoritmi hylkää purkuoptimoinnin mallinnusvirheen aiheuttaman kustannuspoikkeaman vuoksi.

Minuutin laskenta on tarkoitettu käytettäväksi todellisessa tilanteessa silloin, kun ajon aikana kuljettaja huomaa esimerkiksi jonkin reitin osan muuttuneen ajokelvottomaksi. Tällöin muutostieto merkitään kartalle ja uusi laskenta suoritetaan mahdollisimman nopeasti. Tämä versio algoritmista on kuitenkin suunniteltu pidempään laskentaan ja pikalaskentaan kannattaisikin poistaa sadan iteraation välein tapahtuva kauppamatkustajan ongelman ratkaisuun perustuva kokonaislukuoptimointi kokonaan laskennan keskeltä. Vasta lopullisista reiteistä poistettaisiin ylimääräiset silmukat ja edestakaiset liikkeet kyseisellä menetelmällä.

7. YHTEENVETO

Työssä suunniteltiin toistuvien sovitusten menetelmään perustuva kokonaislukuoptimointialgoritmi metsätraktorin korjuureittien suunnittelun automatisoimiseksi. Suunniteltua algoritmia kokeiltiin metsäkoneenkuljettajien ”Taitaja 2014” -kilpailun hakkuualuekartalla. Algoritmilla saatuja tuloksia verrattiin Taitaja-kilpailun finalistien tuloksiin. Tuloksen kehittymistä tutkittiin myös laskennan suorittamien iteraatiokertojen funktiona, jotta nähtäisiin, kuinka nopeasti tulos kehittyy halutulle tasolle.

Vaatimusmäärittelyssä toivottu tunnin laskenta-aika saavutti Taitaja-kartalla lähes finalistien keskiarvoista tulosta vastaavan tuotoksen. PFG:n tuottaman ajosuorituksen loppuraportin perusteella parannettavaa on etenkin ajorajoiteurien yhdistämisessä pääoptimointiratkaisuun. Purkuoptimointi sen sijaan toimi poikkeavasta kuormanmuodostuksesta huolimatta keskitasoisesti Taitaja-finalistien suoritustasoon verrattuna. Kokonaisuutena algoritmi toimi lupaavasti, etenkin otettaessa huomioon kustannusten muodostumisten poikkeamat algoritmin ja PFG:n välillä.

Ensisijainen kehityskohde algoritmille onkin ajorajoitettujen urien liittäminen jo toistuvien sovitusten menetelmän kustannusmatriisiin. Tämän jälkeen algoritmia tulisi voida testata todellisessa kuormatraktorin toimintaympäristössä, jolloin voidaan alkaa säätää sen siirtymäkustannusten muodostumista paremmin todellisuutta vastaavaksi. Ponsse Oyj:n edustajien toiveena oli myös saada algoritmi tuottamaan uusi reittisuunnitelma, jos kuljettaja havaitsee muutoksia ajourien kuljettavuudessa. Tämä ei kuitenkaan vaadi muutoksia toistuvien sovitusten menetelmään vaan ainoastaan sen käyttämään graafiin.

LÄHTEET

[1] H. Lindeman, J. Ala-Ilomäki, M. Sirén, M. Vastaranta, M. Holopainen ja M. Uusitalo, ”Turvemaan kantavuuden ennustaminen laserkeilausaineistolla,” *Metlan työraportteja,* nro 263, 2013.

[2] P. Flisberg, M. Forsberg ja M. Rönnqvist, ”Optimization Based Planning Tools for Routing of Forwarders at Harvest Areas,” *Canadian Journal of Forest Research,* pp. 2153-2163, 25 March 2007.

[3] M. R. Garey ja D. S. Johnson, Computers and intractability - A Guide to the Theory of NP-Completeness, New York: W. H. Freeman and Company, 1979.

[4] P. Wark ja J. Holt, ”A Repeated Matching Heuristic for the Vehicle Routeing Problem,” *The Journal Of the Operational Research Society,* osa/vuosik. 45, nro 10, pp. 1156-1167, October 1994.

[5] O. Ringdahl, Automation in Forestry - Development of Unmanned Forwarders, Umeå: Department of Computing Science, Umeå University, 2011.

[6] K. Kortelainen, ”Kuka teki Suomen ensimmäisen metsäkoneen?,” *Tekniikan historia,* 1 2014.

[7] Ponsse Oyj, ”Luonnonvarassa: Einarin perintö,” [Online]. Available: http://www.luonnonvarassa.fi/artikkelit/einarin-perinto.html. [Haettu 18 Syyskuu 2018].

[8] Ponsse Oyj, ”Ponsse Elephantking,” [Online]. Available: https://www.ponsse.com/fi/tuotteet/kuormatraktorit/elephantking. [Haettu heinäkuu 2018].

[9] S. m. ry., ”sanasto: välivarasto,” [Online]. Available: https://smy.fi/sanasto/valivarasto-intermediate-storage/. [Haettu heinäkuu 2018].

[10] Ponsse Oyj, ”Ponsse: kuormatraktorijärjestelmät,” [Online]. Available: https://www.ponsse.com/fi/tuotteet/opti-tietojaerjestelmaet/metsaekoneen-tietojaerjestelmaet/kuormatraktorit. [Haettu heinäkuu 2018].

[11] R. Ylimäki, K. Väätäinen, S. Lamminen, M. Sirén, J. Ala-Ilomäki, H. Ovaskainen ja A. Asikainen, ”Kuljettajaa opastavien järjestelmien tarve ja hyötypotentiaali koneellisessa puunkorjuussa,” Metsäntutkimuslaitos, Vantaa, 2012.

[12] D. E. W, ”A Note on Two Problems in Connexion with Graphs,” *Numerische Mathematik,* nro 1, pp. 269-271, 1959.

[13] P. Hart E, N. Nils J ja B. Raphael, ”A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions of Systems Science and Cybernetics,* osa/vuosik. 4, nro 2, pp. 100 - 107, 1968.

[14] K. Ruohonen, ”Graafiteoria,” Tampereen teknillinen yliopisto, Tampere, 2013.

[15] G. Laporte, ”Vehicle Routing Problem: an Overview,” *European Journal of Operational Research,* nro 59, pp. 345 - 358, 1992.

[16] P. Eveborn, P. Flisberg ja M. Rönnqvist, ”Laps Care - an Operational System for Staff Planning of Home Care,” *Journal of Operational Research,* nro 171, pp. 962-976, 2006.

[17] G. B. Dantzig ja J. H. Ramser, ”The Truck Dispatching Problem,” *Management Science,* osa/vuosik. 6, nro 1, pp. 80 - 91, 1959.

[18] P. Toth ja D. Vigo, The Vehicle Routing Problem, Society for Industrial and Applied Mathematics, 2002.

[19] B. Zmazek, A. Taranenko ja M. Smid, ”Capacitated VRP with Time Windows and Multiple Trips within Working Day,” 27th, Cavtat, Croatia, 2005.

[20] F. S. Hillier ja G. J. Lieberman, Introduction to Operations Research, 7th toim., McGraw-Hill, 2001.

[21] ”Mixed-Integer Linear Programming Algorithms,” MathWorks, 2018. [Online]. Available: https://se.mathworks.com/help/optim/ug/mixed-integer-linear-programming-algorithms.html#btv20bd. [Haettu 2 syyskuu 2018].

[22] E. D. Andersen ja K. D. Andersen, ”Presolving in Linear Programming,” *Mathematical Programming,* nro 71, pp. 221-245, 1995.

[23] G. B. Dantzig, ”Origins of the Simplex Method,” Systems Optimization Laboratory , Stanford, 1987.

[24] L. G. MItten, ”Branch-and-Bound Methods: General Formulation and Properties,” *Operations Research,* osa/vuosik. 18, nro 1, pp. 22-34, 1970.

[25] M. Forbes, J. Holt, P. Kilby ja A. Watts, ”A Matching Algortihm with Application to Bus Operations,” *Australasian Journal of Combinatorics,* nro 4, pp. 71-88, 1991.

[26] K. Altinkemer ja B. Gavish, ”Parallel Savings Based Heuristics for the Delivery Problem,” *Operations Research,* nro 39, pp. 456-469, 1991.

[27] M. Desrochers ja T. W. Verhoog, ”A New Heuristic for the Fleet Size and Mix Vehicle Routing Problem,” *Computers and Operations Research,* nro 18, pp. 263-274, 1991.

[28] G. Dantzig, R. Fulkerson ja S. Johnson, ”Solution of a Large-Scale Traveling-Salesman Problem,” *Journal of the Operations Research Society of America,* pp. 393-410, 1954.

[29] The Mathworks inc., ”Mathworks documentation: Traveling Salesman Problem: Solver-Based,” [Online]. Available: https://se.mathworks.com/help/optim/ug/travelling-salesman-problem.html. [Haettu 6 syyskuu 2018].

[30] D. A. Spielman, ”Design and Analysis of Algorithms - Lecture 5,” Yale University, 2017.

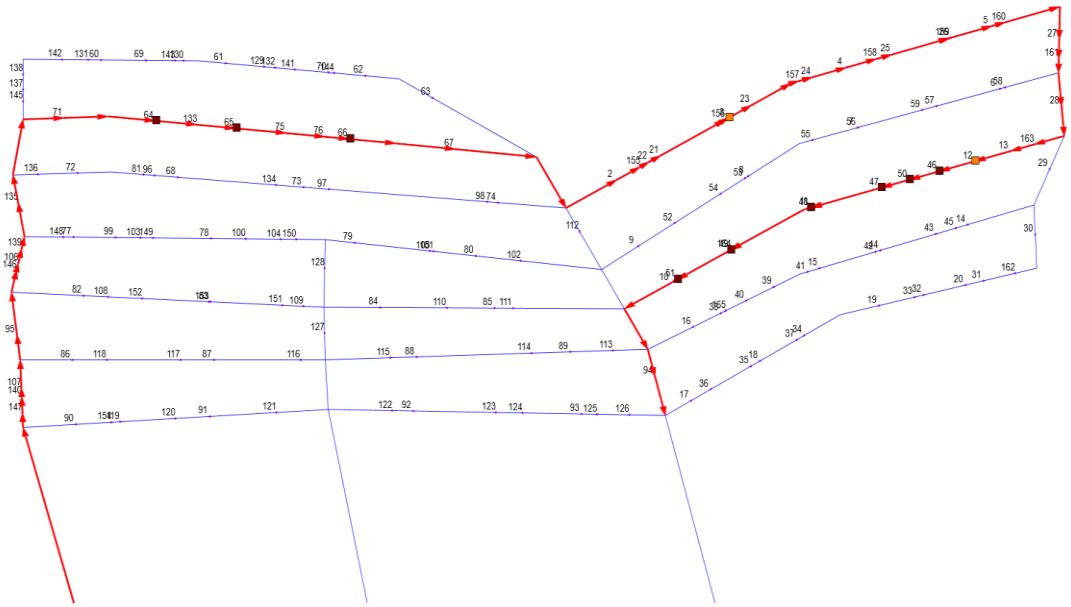
[31] Ponsse Oyj, ”Ponsse Forwarder Game 1.42:n peligrafiikka,” 2017.

LIITE: ESIMERKKI ALGORITMIN KÄYTÖSTÄ PONSSE FORWARD GAMEN KANSSA

Algoritmi tuottaa reittiratkaisun matriisimuodossa:

```
1  64  65  66   3  12  46  50  47  48  49  51   0   0   0   0   0   0
1  72  60  69  61  70  62  21  29  20  32  19  34  37  35  36  17   0
1  67  22  58   6  55   8  53  54  23  25  28  14  45  31  33  18   0
1  92 122 121  91 119  90 118 117  87 116  78 100 105 101  80 102  0
1 120 154 147 106 148  99 103 149 112  98 113 114 126 125 124 123   0
1 150 151 153  71 145  74   2 155 156 159   5 163  13  11 164  10   0
1  93 111  85 110  84  88 115 127 104  79 128 109  83 108  86 107   0
1 133  75  76 132 129 130 143 131 142 138 137 136 135 139 140   0   0
1  95  82 152 146  77 141 144 157 158 160 161 162 165  89  94   0   0
1  81  96  97  57  59  56   7  52   9  63  24   4  26  27  15  16   0
1  68 134  73  30  43  44  42  41  39  40  38   0   0   0   0   0   0
```

Josta tuotetaan ajo-opaste kartalle (esimerkkinä matriisin ensimmäinen reitti eli ensimmäinen rivi). Tämä on nähtävissä kuvassa 15.



Kuva 15. *Algoritmin ratkaisusta tuotettu ajo-opaste kerättävine pinoineen.*

Tämä tuottaa Ponsse Forwarder Game:n raporttiin kuvassa 16 nähtävän reitin, jossa vihreä väri edustaa tyhjänä ajoa, vaaleanpunainen kuormausajoa ja keltainen ajoa viimeisen pinon keräämisen jälkeen purkupaikalle.

Kuva 16. *PFG:n tuottaman raportin kuvaus ajetusta reitistä. Kerätyt pinot vahvennettuina.*

PFG:n raportti tuottaa myös taulukossa 1 nähdyt tiedot ajon tuottavuudesta ja suoritusajoista.